Universidad Autónoma de Madrid
Escuela politécnica superior

Doble Grado en Ingeniería Informática y Matemáticas

Trabajo Fin de Grado

FILTRADO COLABORATIVO EN ESPACIOS TOPOLÓGICOS

Autor: Jesús Serrano Priego
Tutor: Fernando Díez Rubio

JULIO 2017

FILTRADO COLABORATIVO EN ESPACIOS TOPOLÓGICOS

Autor: Jesús Serrano Priego
Tutor: Fernando Díez Rubio

Departamento de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
JULIO 2017

Resumen

Resumen

Con el auge de los sistemas de recomendación en aplicaciones y páginas web que usamos a diario, se generan ingentes conjuntos de datos derivados de la actividad de los usuarios en dichos sistemas. Además de las numerosas técnicas de análisis de datos existentes, se está empleando el Topological Data Analysis (TDA) como una herramienta novedosa más. La confluencia de estos dos ámbitos conjuntamente creemos que no se ha realizado hasta el momento. El objetivo de este trabajo es aplicar esta teoría para medir similitudes entre usuarios en un sistema de recomendación que use filtrado colaborativo. Para ello, es necesario entender cómo transformar el problema de recomendación a los requisitos del TDA. El método de medir similitudes que se propone se basa en la topología algebraica y una de sus formas de implementación, mediante una representación con códigos de barras. Cada usuario en el sistema tiene una representación en forma del código de barras mencionado. De esta forma, podemos comparar usuarios entre sí mediante un algoritmo que usa esta representación topológica. Para poder probar la eficacia de este método, se ha desarrollado un sistema de recomendación en el que poder probar la similitud, conjuntamente con otras similitudes usuales, usando el conjunto de datos 'MovieLens 100k'. Aunque las diferentes pruebas realizadas no mejoran el resultado que se tienen con algunas medidas de similitud usuales, con este trabajo se ofrece un nuevo enfoque al TDA en el problema de la recomendación que, hasta donde conocemos, nunca se había abordado. Propone varios trabajos futuros a desarrollar para realizar mejoras en el método propuesto que pueden dar resultados prometedores.

Palabras Clave

Sistemas de Recomendación, Filtrado Colaborativo, Similitudes, Topological Data Analysis, Topología Algebraica, Códigos de Barras.

Abstract

The rise of the recommendation systems due to applications and web pages that we use every day, results in huge data sets derived from the activity of the users in those systems. In addition to the numerous existing data analysis techniques, the Topological Data Analysis (TDA) is being used as a new tool. We believe that the confluence of these two areas together has not been done so far. The objective of this work is to apply this theory to measure similarities among users in a recommendation system that uses collaborative filtering. For this, it is necessary to understand how to transform the problem of recommendation to the requirements of TDA. The method of measuring similarities proposed is based on the algebraic topology and one of its forms of implementation, using bar codes. Each user in the system has a representation in the form of the mentioned bar code. In this way, we can compare users with each other using an algorithm that uses this topological representation. In order to test the accuracy of this method, a recommendation system has been developed to test the similarity, as well as other usual similarities, using the data set 'MovieLens 100k'. Although the different tests performed do not improve the result of some usual measures of similarity, this work offers a new approach to TDA in the problem of recommendation, which, as far as we know, has never been addressed. It proposes several future works to be developed to make improvements in the proposed method which can yield promising results.

Key words

Recommender System, Collaborative Filtering, Similarities, Topological Data Analysis, Algebraic Topology, Bar Codes.

Agradecimientos

A Fernando por ser la persona que me ha ayudado durante este trabajo, y por haberme animado en los momentos más duros en este camino.

A mis amigos y compañeros por estar ahí siempre.

Y a mis padres por apoyarme y hacer que todo esto sea posible.

Índice general

Índice de Figuras

Índice de Tablas

1

Introducción y motivación del proyecto

Los **sistemas de recomendación** son una pieza fundamental de muchas de las redes sociales y sitios de compra online que existen hoy en día. Estos sistemas recopilan todo tipo de información sobre los usuarios que interactúan con el sitio web, con el fín de poder recomendarles artículos que puedan ser de su interés. Generalmente, los sistemas de recomendación comparan el perfil de un usuario con algunas características de referencia, e intentan predecir la puntuación que el usuario daría a los artículos que aún no ha considerado.

Con el auge de estos sistemas, se estudian diferentes maneras de tratar el conjunto de datos del que se dispone para poder realizar con éxito la tarea de recomendar. De esta manera, se pueden distinguir varios tipos de sistemas de recomendación. Los sistemas orientados en el filtrado colaborativo recomiendan artículos a los usuarios basándose en los intereses de usuarios similares o de artículos similares que ya hayan puntuado. Los orientados a usuarios establecen similitudes entre ellos para poder compararlos y así poder formar vecindarios que agrupen los usuarios que más se parecen.

Estos grandes conjuntos de datos con los que se trabaja, pueden resultar muy costosos de manejar, por lo que, sería apropiado realizar una simplificación a la información más relevante. El **Topological Data Analysis** es un novedoso enfoque al análisis de conjuntos de datos, que usa técnicas de la topología. El TDA está siendo recientemente utilizado en cartografía, tratamiento de imágenes y en biología, entre otros ámbitos.

Puesto que la topología utiliza una formalización de las matemáticas muy abstracta, para poder llevar a cabo estos métodos, se hace uso de la topología algebraica que tiene la ventaja de ser más fácil de implementar computacionalmente. La motivación inicial del TDA es estudiar la forma de los datos. El primer paso a realizar es ver el conjunto de datos como un espacio topológico en el que poder usar estas técnicas de la topología algebraica. El paso final es obtener una representación que muestre de forma incremental la estructura que tiene el espacio topológico y así poder obtener la información más relevante del conjunto de datos.

Puesto que no se ha encontrado constancia de la existencia de un trabajo similar en el que se aplique el TDA al tratamiento de datos de un sistema de recomendación, surge la motivación de este TFG: implementar un sistema de recomendación que emplee técnicas usadas para el tratamiento de conjuntos de datos según el TDA. La idea de este TFG es estudiar nuevas similitudes que puedan utilizarse en los sistemas de recomendación orientados al filtrado colaborativo entre usuarios, utilizando la metodología del TDA. Tras esto, se implementará un sistema de

recomendación para poder probar esta similitud y compararla con las ya existentes, y así poder comparar resultados.

1.1. Objetivos del proyecto

Los objetivos presentes en la realización de este trabajo son los siguientes:

1. Investigación de la metodología que sigue el TDA para el procesamiento de conjuntos de datos.
2. Implementación de una similitud que utilice técnicas del TDA para comparar usuarios.
3. Diseño e implementación de un sistema de recomendación orientado al filtrado colaborativo, así como de otras similitudes ya existentes, como la del coseno o la de Pearson, con el objetivo de usarlas para comparar.
4. Aplicar el sistema de recomendación al conjunto de datos Movielens 100k [1].
5. Comparar los resultados obtenidos de diferentes métricas de errores, como el RMSE y el MAE, sobre las similitudes implementadas.

Para la realización de este trabajo se han utilizado conocimientos aprendidos en diversas asignaturas que se han cursado a lo largo del doble Grado:

1. **Cálculo numérico** y **Métodos numéricos para EDO**. Aprendizaje de Matlab.
2. **Topología**. Conceptos básicos sobre topología que se usan en este trabajo.
3. **Estructuras algebraicas**, **Álgebra conmutativa** y **Teoría de Galois**. Conceptos sobre topología algebraica y homología que se usan en este trabajo.
4. **Análisis** y **diseño de software**. Realización del diseño del sistema de recomendación.
5. **Análisis de algoritmos**. Implementación de algoritmos para las similitudes usadas en el sistema, así como la estimación de su eficiencia.
6. **Inteligencia artificial**. Desarrollo de un sistema que aprende con un conjunto de entrenamiento, para después recomendar items en base a lo aprendido.

1.2. Estructura de la memoria

Este documento se divide en 6 capítulos. El primero se corresponde con esta introducción. El **segundo capítulo** muestra el estado del arte sobre la topología computacional, mostrando los avances que se han realizado en esta materia, y en qué otros ámbitos se ha hecho uso de ella. En el **tercer capítulo** se presentan los conceptos teóricos que presentan nuestro marco de trabajo. Para ello, se realiza una explicación de los diferentes sistemas de recomendación según una clasificación, los conceptos teóricos sobre topología y más en concreto, la metodología que sigue el TDA. En el **cuarto capítulo**, se muestra el desarrollo de la similitud implementada, así como el sistema de recomendación sobre el que se va a trabajar. En el **quinto capítulo**, se muestran los experimentos realizados para probar nuestra similitud implementada, así como los resultados obtenidos. Y, por último, en el **sexto capítulo** se presentan las conclusiones obtenidas sobre el trabajo, así como los posibles trabajos futuros que han surgido.

2

Estado del arte

En este capítulo se analizarán artículos que hayan aproximado el problema que se va a tratar. Puesto que no se ha encontrado constancia de trabajos en los que se aplique el TDA a los sistemas de recomendación, se estudiarán trabajos de otros ámbitos en los que se haya aplicado la topología computacional para el analisis de conjuntos de datos.

El **TDA** es una subárea de la topología computacional que desarrolla técnicas para el análisis de conjuntos de datos científicos. Estas técnicas están basadas en la topología, que es la rama de las matemáticas que estudia las propiedades de los espacios cuando se les aplican ciertos tipos de deformaciones. Analicemos los tres conceptos importantes del TDA [2] [3] [4].

La **topología** estudia la forma que tiene un conjunto de puntos con un espacio topológico asociado. Esta forma se conserva cuando realizamos cierto tipo de transformaciones, como traslaciones o rotaciones. Las propiedades que tienen estos conjuntos de puntos que no cambian al aplicar este tipo de transformaciones, se llaman invariantes topológicas. Sin embargo, hay otras transformaciones que sí modifican estas propiedades invariantes. Una de las invariantes que más se estudia en topología es la conexión. Es por ello, por lo que 'cortar' y 'pegar' una curva en un espacio, no es una tranformación que mantiene invariante la conexión del espacio. Estas tranformaciones se muestran en la figura 2.1. La topología, entonces, clasifica pricipalmente la forma de un espacio por la conexión entre sus elementos.

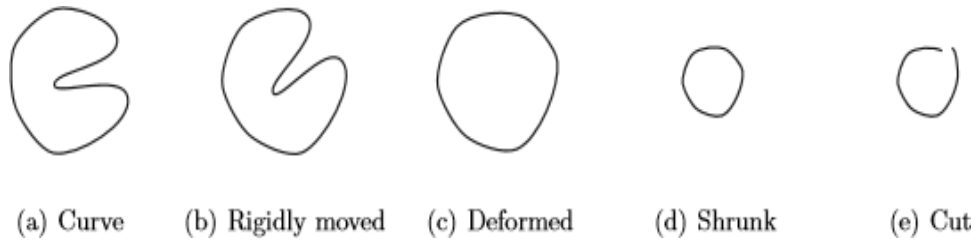


Figura 2.1: Transformaciones topológicas [2]
(a) Curva original. (b) Curva rotada. (c) Curva deformada. (d) Curva deformada y encogida. (e) Curva cortada. Esta última no conserva las invariantes.

Los **datos** a analizar serán, principalmente, procesados de manera computacional y con

[2]Fuente: imagen tomada de [2]

imperfecciones debidas a la manera en que se recolectan. Estos datos pueden ser imágenes bidimensiones de cámaras digitales, imágenes escaneadas de forma tridimensional de partes del cuerpo humano, variedades que simulen la forma que tienen ciertas proteinas en biología, etc.

El **análisis** que se realiza de estos datos, tiene que tener en cuenta las posibles imperfecciones de los datos. Estas imperfecciones surgen a la hora de obtener los datos. En los ejemplos citados anteriormente, en las imágenes obtenidas por la cámara digital, tenemos que tener en cuenta que son imágenes en dos dimensiones pero que captan escenarios reales que están en un espacio tridimensional. Supongamos que tenemos un conjunto de puntos en un espacio d-dimensional. Nuestro ejemplo de datos obtenido de muestra, podría estar en un espacio k-dimensional con $k <= d$, tal y como ocurre en el ejemplo de la imagen bidimensional captada por la cámara digital. Nuestro objetivo en el análisis será la recuperación de información perdida durante la obtención de la muestra, para que los datos sean más realistas.

La metodología del TDA principalmente sigue los siguientes pasos a la hora de analizar un conjunto de datos:

1. Aproximamos el conjunto de datos, visto como un espacio X, a una estructura sólida K, como se muestra en la figura 2.2 (b).
2. Obtenemos las invariantes topológicas de K, como el ciclo visto en la figura 2.2 (c). Un invariante usado conmúnmente son los números de Betti.

En posteriores capítulos se explican los métodos usados en el TDA, que combinen la implementación de estos dos pasos.

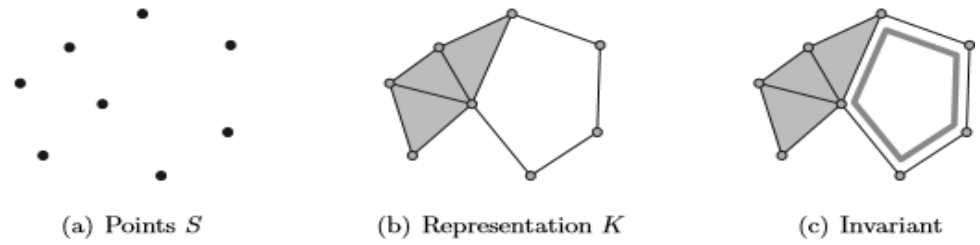


Figura 2.2: Pasos a seguir en el TDA [3]

(a) Representar el conjunto de datos en un espacio. (b) Hallar un complejo simplicial asociado al conjunto de puntos. (c) Aqui se muestra un ciclo que es una invariante topológica, otro ejemplo de invariantes son los números de Betti.

La **topología computacional** se ha aplicado en bastantes ámbitos distintos en el mundo de la ciencia. Algunos ejemplos son el procesamiento de imágenes, la cartografía, el modelado de estructuras y el modelado molecular [5].

En el **procesamiento de imágenes**, se considera una imagen como un rectángulo formado por píxeles. Restringimos nuestra atención a imágenes binarias, en las que los píxeles son solo blancos y negros, donde los negros representan la imagen y los blancos el fondo. Se pueden establecer diferentes tipos de conexiones entre los píxeles, donde dos píxeles están conectados si comparten aristas o, por otro lado, si comparten aristas o vértices. Una técnica común en el procesado de imágenes es reemplazar la imagen por una serie de trazos que representen su esqueleto. Existen diversos algoritmos para obtener el esqueleto de una imagen. Para este proceso puede emplearse la Topología.

Una práctica común en **cartografía** es la combinación de dos mapas que representan información diferente, para poder relacionarla según cada zona. Otra práctica común es la deformación

[3]Fuente: imagen tomada de [2]

de un mapa, en la que salen delimitadas ciertas zonas, con el objetivo de cambiar el área de cada zona en función de una propiedad. Estos mapas se llaman *cartogramas*. Por ejemplo, deformar el mapa de los Estados Unidos en función de los votos existentes en cada Estado. Esto ofrece una mayor información visual sobre lo que se quiere representar. Estas prácticas se realizar utilizando métodos topológicos que deformen las regiones del mapa.

En cuanto al **modelado de estructuras**, surgen tres enfoques de cómo aplicar la topología a este tema: cómo representar una estructura en un ordenador, cómo reconstruir una variedad a partir de un conjunto discreto de datos y cómo extraer propiedades de una estructura. Las propiedades topológicas que pueden extraerse de estas estructuras son los números de Betti.

2.1. Modelado Molecular

El estudio de la forma que tienen las proteínas es un tema importante en biología, ya que puede ayudar a la comprensión de enfermedades y al desarrollo de ciertos medicamentos. Este trabajo se realiza representando cada uno de los átomos de la proteína en un espacio y viendo sus formas o las conexiones entre ellos [6].

Este estudio puede realizarse con un enfoque más geométrico o con uno más topológico. El *enfoque geométrico* se centra más en la forma que tiene la molécula viéndola como una figura tridimensional, sin embargo, el *enfoque topológico* se centra más en las conexiones existentes entre los átomos. Lo interesante de esto, es usar ambos enfoques para realizar un estudio mucho más completo y poder comparar proteínas de manera más precisa, usando solo uno de los dos, se pierde información relevante. Nos centraremos en el estudio de este artículo debido a que es el trabajo que más se aproxima a nuestra comparativa de los usuarios en el sistema de recomendación. Son estudios parecidos, ya que nos centramos en hallar la similitud entre dos objetos de los que extraemos información sobre su estructura.

La *topología* es la rama de las matemáticas que estudia las propiedades que tienen los objetos al deformarlos de forma continua. Las propiedades que se mantienen tras aplicar una de estas transformaciones continuas, se llaman *invariantes*. El estudio de la estructura de la proteína con el enfoque topológico, se va a centrar en las tres siguientes características cuantificables que son invariantes: el número de componentes conexas, el número de agujeros en la estructura y el número de cavidades, esto es, agujeros de dimensión dos. Estos números se llaman *números de Betti*. El cálculo de estos números puede llegar a ser muy complicado en espacios topológicos muy grandes, y más si se realiza de forma computacional, ya que implementar estos conceptos matemáticos en un programa puede ser una tarea difícil. Estos números se van a calcular de forma incremental estableciendo cada vez más conexiones entre los átomos que la forman. Este procedimiento se puede observar en la figura 2.3.

Puesto que hemos calculado los números de Betti para varios radios de esferas de forma incremental, se pueden representar estos números en función del radio usado. Se va a usar una representación especial con códigos de barras. En el eje horizontal se representaría de forma incremental el radio y cada barra representaría *'la línea de vida'* de la componente conexa, del agujero o de la cavidad. En la figura 2.4 se muestra un ejemplo de código de barras de una proteína.

En este artículo se presenta un algoritmo que sirve para comparar códigos de barras, utilizando el coeficiente de Jaccard para cada una de las barras dos a dos [6]. Este es el algoritmo que se usa para establecer similitudes entre proteínas y es el que vamos a utilizar nosotros para comparar los usuarios.

[4]Fuente: imagen tomada de [6]
[5]Fuente: imagen tomada de [6]

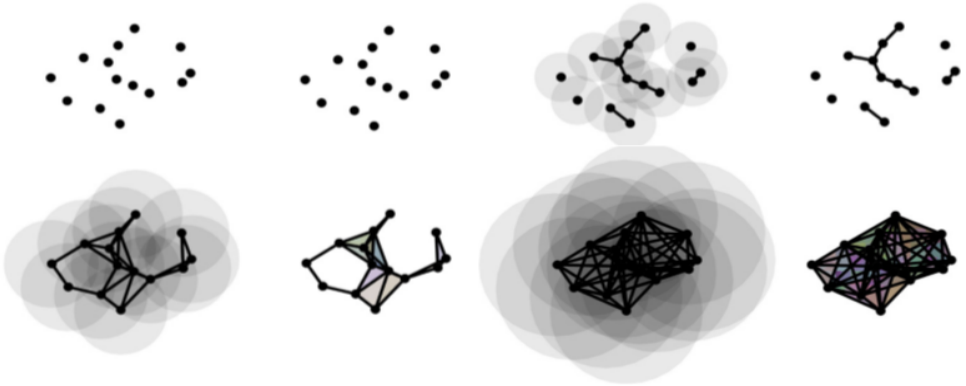


Figura 2.3: Complejos Simpliciales de las proteinas [4]

Se puede observar paso a paso cómo se van construyendo los diferentes complejos simpliciales aumentando el radio de las circunferencias de cada punto. Primero se muestran las circunferencias que se generan y después las conexiones resultantes que éstas establecen.

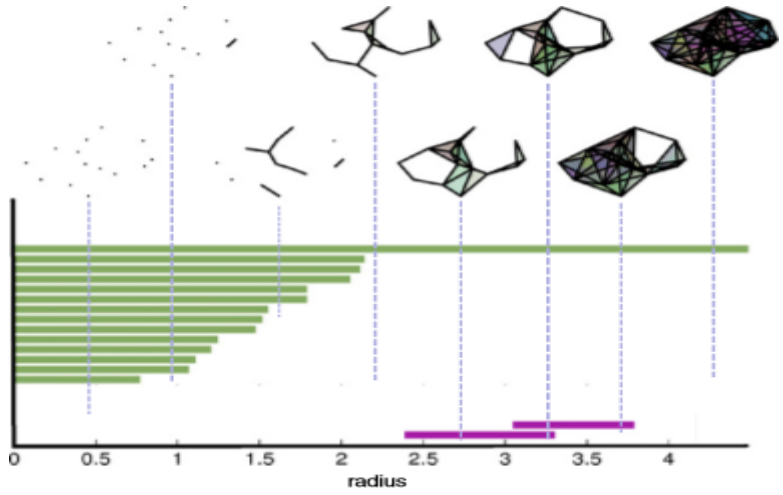


Figura 2.4: Código de barras de una proteína [5]

Código de barras para un conjunto concreto de puntos en dos dimensiones. El eje horizontal representa el radio de las circunferencias crecientes. Las líneas verdes corresponden a las componentes conexas y las moradas a los agujeros de dimensión 1. Cada barra representa la línea de vida en función del radio de los números de Betti. En la parte de arriba, se puede observar cómo evoluciona el complejo simplicial para ayudar a la visalización de cada una de las barras. Cabe resaltar, que a medida que el radio aumenta, el número de componentes conexas disminuye, hasta que en radio 2.2 solamente hay una componente conexa.

3

Conceptos Teóricos Básicos

En esta sección se presenta la teoría del TDA (Topological Data Analysis). En particular, veremos cómo se aplica a los problemas que resuelve computacionalmente, los conceptos y teoría matemática que hay detrás y las diversas aplicaciones en otros ámbitos científicos donde se llevan a cabo estas técnicas. Puesto que nuestro objetivo es aplicarlo a los sistemas de recomendación, también veremos los distintos tipos que hay y en cuál nos centraremos más en nuestro trabajo.

3.1. Sistemas de Recomendación.

Los **sistemas de recomendación** son aplicaciones software que generan sugerencias de items que puede consumir un usuario. Estas sugerencias están relacionadas con ciertas decisiones que tienen que tomar los usuarios, como qué objeto comprar en una tienda online, qué música escuchar o a qué persona seguir en twitter. El objetivo del sistema de recomendación es hacer uso de un algoritmo con el fin de hallar el item más apropiado para un usuario dependiendo de sus intereses y del contexto de la situación. El término 'item' denota el objeto o producto que el sistema recomienda a los usuarios. Pueden ser muy diferentes: desde una película hasta una persona en una red social [7]. En la figura 3.1 se muestra un ejemplo de matriz de ratings que usa un sistema.

Los sistemas de recomendación están presentes en multitud de sitios web que la gente usa a diario. Algunos ejemplos de ellos son Amazon, YouTube, Netflix, Yahoo, Tripadvisor, Facebook, etc.

Existen varios tipos de sistemas de recomendación. Una clasificación común, como la que se establece en [8], puede ser la siguiente.

3.1.1. Basados en contenido

Estos sistemas recomiendan items a los usuarios basándose en los artículos que le han gustado al usuario en el pasado, es decir, los que ha puntuado positivamente. Esto es, recomendará items que sean similares a otros que el usuario ya ha puntuado. Una desventaja de este sistema es el **encasillamiento** de los gustos, ya que siempre va a recomendar a un usuarios artículos muy

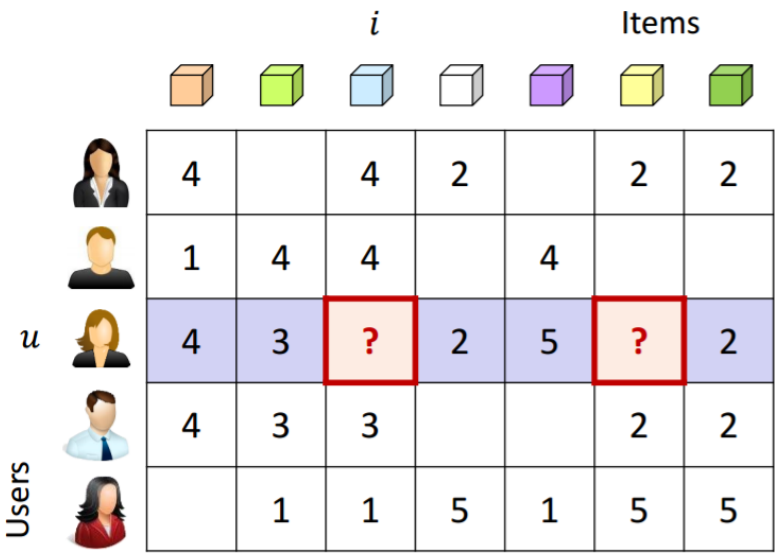


Figura 3.1: Matriz de ratings del sistema [1]

Una matriz de ratings o de puntuaciones tiene como filas los diferentes usuarios del sistema y como columnas los items existentes. Cada número es la puntuación que le ha dado un usuario a un item, en este caso con valores enteros entre 1 y 5. Los huecos vacios son items que no ha puntuado. El objetivo del sistema será predecir la puntuación que le daría el usuario u a los distintos items que no ha puntuado, para después recomendarle uno de ellos.

parecidos entre sí, pero es posible que le puedan interesar otros que sean diferentes.

3.1.2. Filtrado colaborativo

Los sistemas de recomendación basados en **filtrado colaborativo** basan sus recomendaciones a un usuario en los items que otros usuarios similares han puntuado positivamente. Estos sistemas tratan de formar vecindarios de usuarios similares.

Hay otros sistemas de filtrado colaborativo que recomiendan items similares a otros items ya puntuados positivamente por el usuario. Es decir, en este caso se buscan similitudes entre items y, por tanto, se forman vecindarios entre items en lugar de entre usuarios.

Para calcular la similitud entre usuarios o entre items se usan diferentes medidas de similitudes. Las más usadas son la de *Pearson* y la del *coseno*.

Un algoritmo empleado con frecuencia por este tipo de sistemas es el algoritmo de **vecinos más próximos**. Este algoritmo trabaja sobre un agrupamiento de los usuarios en vecindarios. En estos vecindarios, solamente se tienen en cuenta las puntuaciones de los usuarios más similares a éste. Estos métodos de k vecinos más próximos son bastante usados debido a su simplicidad a la hora de ser implementados, a su eficiencia, y a su capacidad para establecer recomendaciones apropiadas.

Tenemos un mayor interés en las características de estos sistemas ya que son en los que vamos a aplicar la nueva similitud basada en espacios topológicos. Por ello, vamos a ver más en detalle sus puntos fuertes y sus desventajas.

Como **puntos fuertes** cabe destacar la **diversidad** en las recomendaciones. A un usuario se le pueden recomendar items que no se aproximen mucho a otros que ya ha puntuado en el pasado, debido a que sus usuarios más similares pueden haber puntuado positivamente items que

[1] Fuente: imagen tomada de apuntes de la asignatura BMIN

no sean parecidos. Esto es una ventaja frente a los sistemas basados en contenido, por ejemplo, ya que éstos recomiendan cosas basándose en items ya puntuados, lo que hace que no hay mucha diversidad.

Por otro lado, una de las principales **desventajas** de estos sistemas es el llamado **arranque frío**. Este fenómeno ocurre cuando un usuario nuevo se registra en el sistema, o se da de alta un nuevo item. Para el caso del usuario nuevo, la tarea de recomendarle items va a ser complicada siguiendo el algoritmo de este sistema, ya que al ser nuevo, la tarea de encontrar usuarios similares no será trivial. De la misma manera, para los items nuevos que se dan de alta en el sistema, nadie los habrá puntuado, por lo que a un usuario no se le pueden recomendar items que nadie ha valorado.

Otra desventaja de estos sistemas es la **escalabilidad**. Hoy en día, se manipulan inmensas cantidades de datos en los sistemas de recomendación, por lo que la eficiencia del algoritmo es muy importante. Sin embargo, un algoritmo de filtrado colaborativo con complejidad O(N) ya es demasiado grande para el tamaño de los datos actuales. Además, muchos de los sistemas tienen que reaccionar a tiempo real para realizar recomendaciones en línea, por lo que se exige una mejor escalabilidad.

3.1.3. Demográficos

Los sistemas de recomendación demográficos tienen en cuenta las características demográficas de los usuarios a la hora de recomendar, como la ubicación geográfica donde vive, el idioma que habla, su sexo, su edad, etc. De esta manera, si tienes 16 años, este sistema te recomendará canciones que puedan interesar a los adolescentes. Normalmente, los buscadores web utilizan este tipo de recomendadores, para ofrecer a los usuarios páginas que estén en su idioma.

3.1.4. Basados en conocimiento

Los sistemas de recomendación basados en conocimiento obtienen información contextual de los usuarios en una situación determinada. De este modo, se tienen en cuenta cómo ciertas características de los items son compatibles con las necesidades de los usuarios, sus preferencias y, en última instancia, cuán útil es el item para el usuario.

En estos recomendadores, la función de similitud que se usa estima si son compatibles las recomendaciones que se van a dar con las necesidades del usuario. Existen dos tipos:

1. Los basados en casos determinan las recomendaciones basándose en medidas de similitud.
2. Los basados en restricciones recomiendan explotando conocimientos predefinidos que contienen reglas sobre cómo relacionar los requisitos del usuario con las características del item a recomendar.

Inicialmente, estos sistemas dan mejor resultado que los demás tipos, pero a la larga tienden a ser peores si no disponen de un módulo de aprendizaje del comportamiento que tienen los usuarios con el sistema.

3.1.5. Basados en comunidades

Estos sistemas de recomendación se basan en las preferencias de los amigos de un usuario a la hora de recomendar un item. Este sistema sostiene que las personas tienden a relacionarse y a entablar amistades con otras personas con las que comparten gustos, o por otro lado, que los

usuarios tienden a interesarse por items que han interesado a sus amigos. La función de similitud de estos sistemas se basa en los ratings que han dado los amigos del usuario. Estos datos son adquiridos de una red social.

3.1.6. Híbridos o mixtos

Los sistemas de recomendación híbridos son una combinación de los sistemas vistos anteriormente. Además, están combinados de manera que las desventajas de uno intenten ser arregladas con las ventajas del otro.

3.2. Teoría de la Topología.

La *Topología* es la rama de las matemáticas que estudia las propiedades de los llamados espacios topológicos, que permanecen inalteradas bajo tranformaciones o deformaciones continuas. Por su parte, la teoría del TDA tiene su fundamento teórico en la topología, por lo que para poder describir cómo se emplea en este contexto, hemos de comenzar introduciendo algunos conceptos teóricos necesarios [9].

3.2.1. Topología

Una **Topología** T sobre un conjunto X es un subconjunto de su conjunto de partes, tal que:

1. Si $S_1, S_2 \in T$, entonces $S_1 \cap S_2 \in T$.
2. Si $\{S_j | j \in J\} \subset T$, entonces $\bigcup_{j \in J} S_j \in T$.
3. $\varnothing, X \in T$.

El par (X, T) es un **Espacio Topológico**.

Ejemplo: Sea el conjunto $X = \{1,\ 2,\ 3\}$. El conjunto de partes de este conjunto es $P(X) = \{\varnothing,\ \{1\},\ \{2\},\ \{3\},\ \{1,\ 2\},\ \{1,\ 3\},\ \{2,\ 3\},\ X\}$. Una topología T sobre el conjunto X, que es un subconjunto de $P(X)$, podría ser $T_1 = \{\varnothing, \{1\},\ \{1,\ 2\},\ X\}$. Pero no podría ser $T_2 = \{\varnothing,\ \{1\},\ \{1,\ 2\},\ \{2,\ 3\},\ X\}$, ya que $\{1,\ 2\} \cap \{2,\ 3\} = \{2\}$, que no está en T. La topología $T_3 = \{\varnothing,\ X\}$ y $T_4 = P(X)$, siempre son dos topologías sobre cualquier conjunto X; son la topología trivial y la discreta, respectivamente.

3.2.2. n-Simplex

Un **n-simplex** x es generado por un conjunto de puntos geométricamente independientes $\{a_0, ..., a_n\}$ en $\mathbb{R}^N$, tal que

$$x = \sum_{i=0}^{n} t_i a_i,\ \text{donde}\ \sum_{i=0}^{n} t_i = 1$$

Por comodidad, se denominarán a los n-simplex simplemente como simplex, ya que la n representa el número de puntos que lo generan.

1. Un conjunto de puntos $\{a_0, ..., a_n\}$ se dice **geométricamente independiente** si para todo número real t_i, las ecuaciones

$$\sum_{i=0}^{n} t_i = 0 \text{ y } \sum_{i=0}^{n} t_i * a_i = 0$$

implican que $t_0 = t_1 = ... = t_n = 0$.

2. Sea x el n-simplex generado por $\{a_0, ..., a_n\}$, se denomina **cara** de x a cada uno de los simplex generados por subconjuntos de $\{a_0, ..., a_n\}$.

3.2.3. Complejo Simplicial

Un **Complejo Simplicial** K en $\mathbb{R}^N$ es una colección de simplex en $\mathbb{R}^N$ tal que:

1. Cada cara de un simplex de K está en K.
2. La intersección de cualesquiera dos simplex de K es una cara de cada uno de ellos, y por tanto, está en K.

En la figura 3.2 se muestra un ejemplo de Complejo Simplicial.

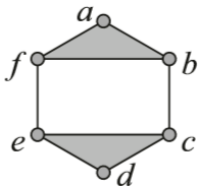


Figura 3.2: Ejemplo de Complejo Simplicial [2]

Cada punto del conjunto es un 0-simplex. Cada arista son 1-simplex. Cada triángulo sombreado es un 2-simplex. El cuadrado blanco del centro es un agujero de dimensión 1.

3.2.4. Nerve

Sea X un conjunto de puntos, y sea (Y, T) un espacio topológico. Sea S un subconjunto de puntos de X. Un **recubrimiento** de S es

$$U = \{U_i\}_{i \in I},\ U_i \subset Y,\ \text{con } U_i \text{ abierto}$$

donde I es un conjunto de índices, y tenemos $S \subset \bigcup_{i \in I} U_i$.

Llamamos **Nerve** N de U al conjunto que cumple

1. $\varnothing \in N$,
2. Si $\bigcap_{j \in J} U_j \neq \varnothing$ para $J \subset I$, entonces $J \in N$.

En consecuencia, el Nerve es un Complejo Simplicial. En la figura 3.3 se muestra un ejemplo de recubrimiento y Nerve.

[2] Fuente: imagen tomada de [2]

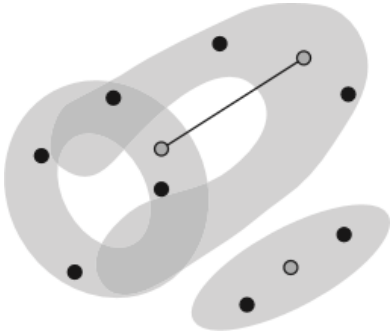


Figura 3.3: Recubrimiento y Nerve [3]

Las zonas sombreadas son los diferentes recubrimientos del conjunto de puntos. Los puntos no rellenados y sus conexiones entre ellos, representan el Nerve del conjunto, que es un Complejo Simplicial

3.2.5. Cech Complex y Vietoris-Rips Complex

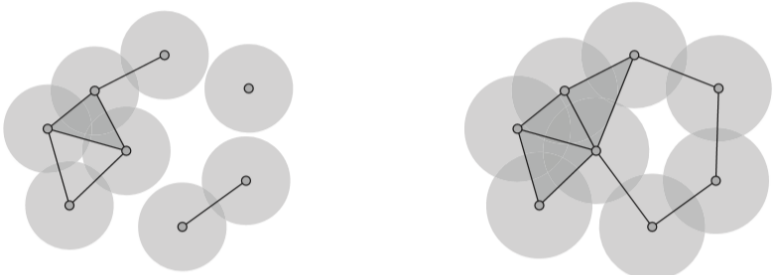
Sea $B_\epsilon(x)$ la bola abierta de radio ϵ y centrada en x. Esto es, para $\epsilon \in \mathbb{R}$ y $x \in Y$,

$$B_\epsilon(x) = \{y \in Y | d(x, y) < \epsilon\}$$

Sea $S \subset Y$ y $\epsilon \in \mathbb{R}$, centramos una bola de radio ϵ en cada punto para obtener un recubrimiento:

$$U_\epsilon = \{B_\epsilon(x) | x \in S\}$$

El Nerve de este recubrimiento es el denominado **Cech Complex** C_ϵ. En la figura 3.4 se muestra un ejemplo de Cech Complex.



(a) C_{ϵ_1} (b) C_{ϵ_2}, $\epsilon_2 > \epsilon_1$

Figura 3.4: Cech Complex de un conjunto de puntos [4]

(a) Hallamos el complejo simplicial C_{ϵ_1} para un radio de circunferencias ϵ_1. (b) Hallamos otro complejo simplicial C_{ϵ_2}, para un $\epsilon_2 > \epsilon_1$

Por otro lado, el Vietoris-Rips Complex es más utilizado debido a la facilidad de su contrucción, incluso en muchas dimensiones. Este complejo se basa en un grafo en lugar de en un Nerve. Sea $S \subset Y$ y $\epsilon \in \mathbb{R}$, sea $G_\epsilon = (S, E_\epsilon)$ el $\epsilon - neighborhoodgraph$ en S, donde

$$E_\epsilon = \{\{u, v\} | d(u, v) \leq \epsilon, u \neq v \in S\}$$

[3] Fuente: imagen tomada de [2]
[4] Fuente: imagen tomada de [2]

Sea un **clique** en el grafo un subconjunto de vértices que induce un **subgrafo completo**, es decir, un subgrafo en el que todos los vértices están conectados entre sí a través de aristas. Un **clique es maximal**, si no existe un clique más grande que lo contenga. El **Clique Complex** es el complejo simplicial formado por los cliques maximales de un grafo. El **Vietoris-Rips Complex** es el Clique Complex de G_ϵ.

En nuestro trabajo, se va a hacer uso de Vietoris-Rips Complex, ya que son los más eficientes de calcular.

(a) Graph & Cliques (b) VR complex $V_{2\epsilon_1}$

Figura 3.5: Vietoris-Rips Complex de un conjunto de puntos [5]

(a) $\epsilon - neighborhood$ graph para un ϵ fijo, y sus cliques. (b) Vietoris-Rips Complex asociado a los cliques del grafo.

3.2.6. Homeomorfismos e invariantes

Sean (X, T_1) e (Y, T_2) dos espacios topológicos. Una **función continua** f cumple que $\forall V \in T_2$, tenemos que $f^{-1}(V) \in T_1$.

Sean (X, T_1) e (Y, T_2) dos espacios topológicos y sea f una función de X a Y; entonces, f es un **homeomorfismo** si se cumple que:

1. f es una biyección
2. f es continua
3. f tiene inversa continua

Dos espacios se dicen **homeomorfos**, si existe un homeomorfismo entre ellos. Como ya se ha mencionado antes, un homeomorfismo puede ser una función que deforma, traslada o rota el espacio, pero no puede realizar cortes en la figura.

Las propiedades **invariantes** de un espacio topológico son las que no cambian al aplicar un homeomorfismo.

Sea K cualquier Complejo Simplicial, la **característica de Euler** $\chi(K)$ es

$$\chi(K) = \sum_{\sigma \in K} (-1)^{dim\sigma} = \sum_{n=0}^{dimK} (-1)^n c_n$$

donde c_n es el número de celdas n-dimensionales de K. La característica de Euler es una invariante topológica.

[5] Fuente: imagen tomada de [2]

3.2.7. Grupo de Homología

Sea K un Complejo Simplicial, supongamos que tenemos un orden para su conjunto de vértices. Una **orientación** de un n-simplex $\sigma = \{v_0, v_1, ..., v_n\} \in K$, es una clase de equivalencia en el orden de sus vértices, donde $(v_0, v_1, ..., v_n)$ $(v_{\tau(0)}, v_{\tau(1)}, ..., v_{\tau(n)})$ si la paridad de τ es par. Un **n-simplex orientado** es un simplex con orientación, denotado como la secuencia $[\sigma]$ El **n-ésimo chain group** $C_n(K)$ de K es el grupo abeliano de los conjuntos de K de n-simplex orientados, bajo la relación de equivalencia descrita.

Un elemento $c \in C_n$ es una **n-chain**, $c = \sum_i c_i[\sigma]$, con n-simplex $\sigma_i \in K$ y coeficientes $c_i \in \mathbb{Z}$.

Dada una chain c, el **homomorfismo frontera** $\partial_n : C_n \longrightarrow C_{n-1}$ definido por:

$$\partial_n[v_0, ..., v_n] = \sum_i (-1)^i [v_0, ..., \hat{v_i}, ..., v_n]$$

donde $\hat{v_i}$ indica que v_i no aparece. También definimos $\partial_0 = 0$. Una propiedad fundamental de este homomorfismo es que $\partial_n \circ \partial_{n+1} = 0$ para todo $n >= 0$. El operador frontera conecta la cadena de grupos para ser una cadena de complejos C_*:

$$... \longrightarrow C_{n+1} \longrightarrow C_n \longrightarrow C_{n-1} \longrightarrow ...$$

Para cualquier cadena de complejos, el **n-ésimo grupo de homología** H_n es

$$H_n = ker\partial_n / im\partial_{n+1}$$

donde **ker** e **im** son el núcleo y la imágen del operador, respectívamente. En la imagen de un homomorfismo ∂_{n+1} están los ciclos dados por los (n+1)-simplex del grupo C_{n+1}. Mientras que en el núcleo de ∂_n sólo estarán los ciclos que no son n-simplex en C_n.

Los grupos de homología de un espacio permanecen invariantes bajo homeomorfismos.

Ejemplo: Partimos de la figura 3.2, con el orden alfabético en los vértices del complejo $abcdef$. Sea el 2-simplex abf, al aplicar el homomorfismo ∂_2, nos queda $\partial_2(abf) = bf - af + bf$ que nos da como resultado 0 ya que es un ciclo en C_1. Sin embargo, el ciclo $bc + ce + ef + fb$, no tiene pre-imagen en C_2, ya que no es un 2-simplex, sin embargo, éste pertenece al núcleo de ∂_1 ya que es un ciclo en C_1. De esta manera, éste es el elemento que está en H_1, que salvo equivalencias en la orientación de los vértices, es el único agujero de dimensión 1.

3.2.8. Números de Betti

Sea R un anillo con unidad, sea M un grupo abeliano, entonces M es un **R-Módulo** si existe una operación $\phi : R \times M \longrightarrow M$, que define la operación de multiplicar elementos de R por elementos de M (y quedarse en M), tal que para cualesquiera $r, s \in \mathbb{R}$, $y \in M$, se tiene

1. (rs)x = r(sx)
2. (r + s)x = rx + sx
3. r(x + y) = rx + ry
4. 1·x = x

Un A-Módulo M es **libre**, si M es isomorfo a una suma directa de copias de A. El **rango de M** es el número de copias de A.

El n-ésimo grupo de homología H_n de un complejo simplicial es un grupo. Además es un $\mathbb{Z}$-Módulo. Como el complejo simplicial sobre el que tratamos es finito, H_n es un $\mathbb{Z}$-Módulo finitamente generado. El **n-ésimo número de Betti** es el rango del grupo de homología H_n, visto como $\mathbb{Z}$-Módulo. El primer número de Betti representa el número de componentes conexas del complejo simplicial, el segundo representa el número de agujeros de dimensión uno, el tercero los agujeros de dimensión 2, etc.

Ejemplo: En la figura 3.2, tenemos que hay una sóla componente conexa, por tanto el primer número de Betti es 1. Tenemos que hay un agujero de dimensión 1, por tanto el segundo número de Betti es 1. Y no tenemos agujeros de dimensión mayor, por lo que los siguientes número de Betti son todos 0.

3.3. Topological Data Analysis

Dado un conjunto finito de datos S en Y en los que hay ruido, contenidos en un espacio desconocido X, el TDA se centra en recuperar la topología de X, asumiendo que tanto X como Y son espacios topológicos. Para ello, utiliza los conceptos teóricos explicados en la sección anterior, siguiendo una serie de pasos, hasta encontrar los invariantes topológicos de ambos espacios, que son los de nuestro interés. Tal y como se ha introducido en el capítulo 2, la metodología del TDA sigue dos pasos fundamentales: aproximar nuesto conjunto de datos a una estructura combinacional, obteniendo posteriormente las invariantes topológicas que describen el conjunto.

Puesto que estos datos han de ser tratados de forma computacional, hay que buscar técnicas que sean fáciles de implementar en un ordenador y eficientes. La manera más fácil de implementar estas técnicas en un ordenador es haciendo uso de la topología algebraica.

El paso previo a realizar, es ver el conjunto de datos como un espacio topológico. Para ello, se obtiene un complejo simplicial asociado al conjunto de datos. Hay muchos algoritmos distintos que persiguen conseguir un complejo simplicial asociado a un conjunto de datos. Estos procedimientos dan nombre a los complejos que se obtienen, siendo el Cech Complex, el Alpha Complex, Vietoris-Rips Complex, Cubical Complex, etc. El algoritmo más utilizado, por su facilidad de uso, es el Vietoris-Rips Complex.

Esta estructura se construye para hallar los números de Betti asociados a estos complejos. Obtenemos varios complejos del espacio, que varían en funcion de unos parámetros que vamos incrementando. Por ejemplo, en el Vietoris-Rips Complex, se incrementa el ϵ que establece si en el grafo generado, dos puntos están conectados. De esta forma, obtendremos diferentes números de Betti asociados a cada uno de los complejos generados.

Estos números de Betti se obtienen tal y como se ha explicado en los conceptos teóricos del apartado anterior. Aunque, en la práctica se utilizan estos conceptos de una manera más computable. La factorización del grupo de homología en suma directa de $\mathbb{Z}$-Módulos, se puede ver también como la siguiente fórmula:

$$\chi(X) = \sum_n (-1)^n \beta_n$$

Esta es la fórmula de Euler-Poincaré, que establece una suma alternada entre los números de Betti. Donde cada n-ésimo número de Betti quedaría reducido a calcular el número de ciclos en C_n que no tienen preimagen en C_{n+1} por ∂_{n+1}.

Puesto que hemos obtenidos varios números de Betti para cada uno de los complejos, se pueden representar en un código de barras. La explicación de cómo se obtienen estos códigos de barras se puede ver en [10]. En el eje horizontal se representaría de forma incremental el radio y cada barra representaría 'la línea de vida' de la componente conexa, del agujero o de la cavidad. En la figura 3.6 se muestra un ejemplo de código de barras.

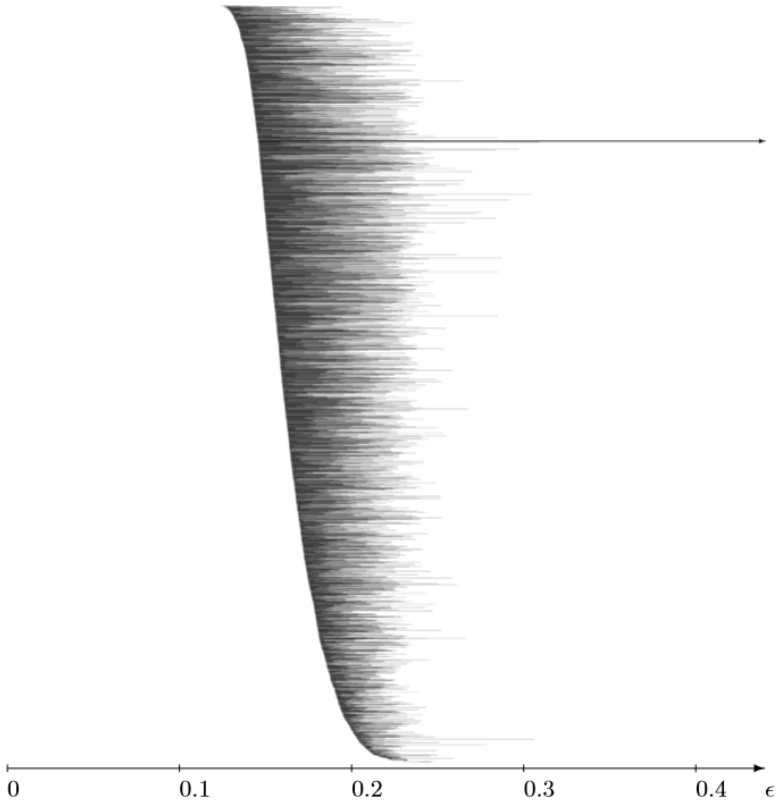


Figura 3.6: Ejemplo de código de barras [6]

Se puede observar que hay una línea que continúa hasta el final (acabada en flecha) que representaría el número de betti para valores muy grandes del radio

[6] Fuente: imagen tomada de [2]

4

Recomendación basada en el TDA

En este capítulo se detalla el desarrollo del sistema que se ha planteado para poder probar la propuesta de recomendación basada en espacios topológicos. En la primera sección, se realiza una explicación de la similitud implementada basada en el TDA. En la segunda sección se describe un análisis del conjunto de datos sobre el que hemos trabajado. Finalmente, en la última sección, se realiza una explicación del diseño y de la implementación del sistema de recomendación con el que se han realizado las pruebas.

4.1. Similitud basada en el Topological Data Analysis

Los sistemas de recomendación usan diversas maneras de medir la similitud tanto entre items como entre usuarios. Este grado de similitud se utiliza para recomendar a un usuario items que se parezcan a otros que ya haya puntuado o para recomendar items que gusten a usuarios similares a él.

La medida de similitud entre usuario o items que usa un sistema de recomendación es solamente una parte de su estructura. La tarea de recomendar un item a un usuario se realiza operando sobre las similitudes entre items o usuarios ya calculadas de antemano, que pueden verse como una matriz de similitudes. Las diferentes similitudes que puede tener un sistema calcula estos valores de diversas maneras. Este es el elemento del sistema de recomendación que nosotros vamos a sustituir por nuestro algoritmo basado en espacios topológicos.

Como ya se ha explicado en las secciones 3.2 y 3.3, a la hora de analizar un conjunto de datos con el TDA, se tiene que cambiar la forma usual que tenemos de ver este tipo de conjuntos sacados de un sistema de recomendación o de otros sitios, y verlo más orientado a ser un espacio topológico. Para este conjunto, que tenemos usuarios, ratings e items; tenemos que intentar verlo como uno de estos espacios. Así que, la primera cuestión que surge es, ¿de qué manera tenemos que interpretar o transformas esta nube de datos para estar en las premisas que impone el TDA para poder ser aplicada su metodología?.

4.1.1. Aplicar el TDA a los datos del sistema de recomendación

El TDA es una teoría que ya ha sido investigada y aplicada en diversos ámbitos científicos en los últimos años, por lo que el uso de sus técnicas tiene un respaldo matemático consistente. Es decir, es una teoría que ya ha sido probada y ha dado sus buenos resultados. Por tanto, probablemente la tarea más importante a llevar a cabo es trasladar el problema de recomendación a las premisas que impone el TDA. Esto es, ¿qué es aquí el espacio topológico sobre el que vamos a estudiar su estructura?

En estos espacios topológicos, a la hora de aplicar las técnicas de homología simplicial explicadas en la sección 3.2 y 3.3, vamos a obtener cómo es su estructura, cuáles son sus números de Betti y cuáles son los códigos de barras asociados a estos espacios. Es decir, al igual que en el artículo del modelado molecular mencionado en la sección 2.1, donde comparaban proteínas entre sí, aquí vamos a comparar usuarios o items entre sí. Por tanto, una primera idea, sería ver cada usuario como un espacio topológico para intentar comparar sus estructuras entre sí. Puesto que la tarea de ver los datos como un espacio topológico se puede enfocar de otras maneras, se pueden estudiar otras opciones en trabajos futuros. En el capítulo 6, donde se explican las conclusiones y trabajos futuros, se explica en mayor profundidad este tema.

Al igual que en las similitudes que se aplican convencionalmente, tenemos que ver a los usuarios como un elemento de un espacio, hablando en términos matemáticos. En muchas similitudes como la del coseno, se ve a los usuarios como un vector en un espacio de dimensión N. Es decir, se puede representar al usuario con un vector. En el conjunto de datos sobre el que trabajamos tenemos varios datos asociados a los usuarios. Entre estos datos, tenemos asociados la lista de items con los que han interactuado y la correspondiente puntuación que le han dado. Si quisiéramos ver a los usuarios como un vector, sus componentes podrían ser los ratings asociados a cada item que han puntuado (del 1 al 5), éstos ordenados por el identificador del item , y ceros en las posiciones de los items que no hayan puntuado.

Pero en nuestro caso, queremos que los ratings que describen los gustos de cada usuario sean enfocados de otra manera, para formar un espacio topológico. Podemos ver los datos en un espacio de dos dimensiones, donde es más visual representar conjuntos de puntos y es más eficiente de implementar. Para ver un usuario como un espacio de dos dimensiones, podemos representar en el eje x los identificadores de los items del sistema, y en el eje y los ratings que han dado a cada objeto, yendo de 1 a 5. De esta forma, si el usuario 1 ha dado una puntuación de 4 al item con identificador 14: habría un punto en las coordenadas (14, 4) en el espacio asociado al usuario 1. Un ejemplo se puede ver en la figura 4.1.

Ahora podemos preguntarnos si este planteamiento está dentro de las premisas del TDA. Tenemos un conjunto de puntos en $\mathbb{R}^2$, sobre el que, al igual que en el ejemplo 1 de la sección 3.2, podemos obtener varias topologías. En la sección 3.2 comentábamos que se puede asociar una estructura sólida a un conjunto de puntos como este. Esta estructura se llama complejo simplicial, y había varias maneras de obtener uno de estos complejos a partir de una nube de puntos. Esta estructura, construida a partir de circunferencias (en este caso) de radio incremental, induce una topología sobre este conjunto. Por tanto, ya tenemos el conjunto de datos enfocado a los procedimientos que vamos a realizar con el TDA.

4.1.2. Desarrollo de la similitud

La similitud implementada utiliza las técnicas del TDA. Tal y como se ha explicado en la sección 3.3, el TDA se centra en el estudio de la estructura que tiene un espacio topológico. En nuestro caso, tenemos espacios que representan a cada usuario del sistema y de los que obtenemos la información relacionada con sus números de Betti.

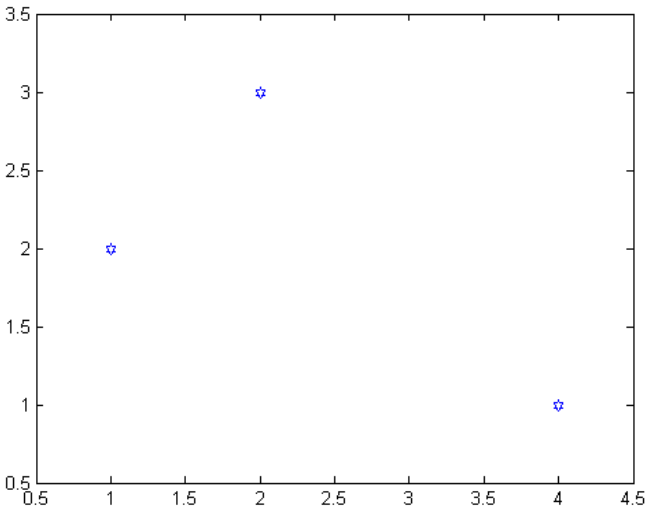


Figura 4.1: Ejemplo de usuario representado

Partiendo de los espacios planteandos en la sección 4.1.1, vamos a obtener una estructura más sólida que represente el conjunto. Esta estructura que estamos buscando es un Complejo Simplicial. En nuestro caso, se obtiene el complejo asociado al Vietoris-Rips Complex. Este complejo simplicial, realiza circunferencias del mismo radio en cada uno de los puntos que representan las puntuaciones de los items de cada usuario. Cuando dos circunferencias se cortan, se traza una arista que une los centros de ambas. La estructura resultante es el Vietoris-Rips Complex. Tras esto, se realiza el cálculo de los números de Betti para el complejo simplicial calculado. Todos estos cálculos se explican en el capítulo 3. Se repite el proceso de obtención del Vietoris-Rips Complex y de los números de Betti asociados para diferentes valores de radios de circunferencia centrados en cada punto. Con esto, se obtiene una evolución de los números de Betti del conjunto de puntos dependiendo del radio escogido. Esta evolución se puede ver en las imágenes 4.2, 4.3 y 4.4

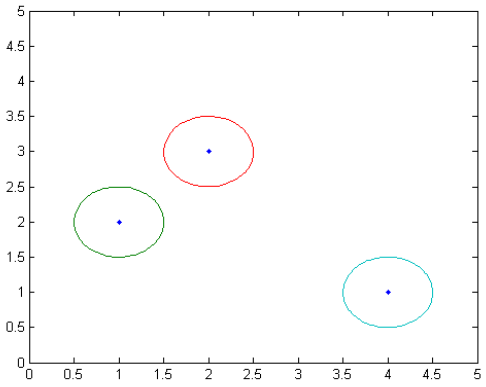


Figura 4.2: Complejo de un usuario para radio 0.5
El complejo asociado al usuario representado en la figura 4.1. Está formado por los 3 puntos únicamente, ya que las circunferencias no tienen un radio suficientemente grande como para contener a otros puntos.

Los grupos de homología con los que se trabaja aquí, que llevan asociados los números de Betti, son componentes invariantes de los espacios topológicos. Esto es, que permanecen

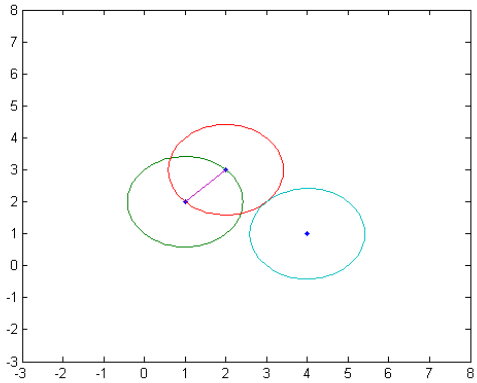


Figura 4.3: Complejo de un usuario para radio igual a raíz de 2
El complejo tiene ahora una arista entre el punto (1, 2) y el (2, 3), ya que las circunferencias de cada una llegan a contener al otro punto.

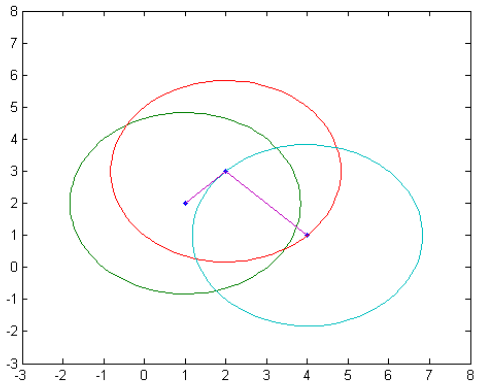


Figura 4.4: Complejo de un usuario para radio igual a raíz de 8
El complejo ya conecta los tres puntos.

inalteradas cuando se aplica una transformación, tal como un homomorfismo. Dos espacios distintos son homeomorfos si tienen las mismas componentes invariantes. Es decir, si dos espacios tienen los mismos grupos de homología, estos dos espacios se dicen homeomorfos. El clásico ejemplo sobre esto es la equivalencia entre una taza y un donut mostrada en la imagen 4.5.

Toda esta información obtenida del espacio de partida, se puede representar para tener la información del espacio de una forma más visual. La información obtenida es una representación de los números de Betti para cada uno de los complejos obtenidos para los diferentes radios escogidos. La representación se realiza mediante códigos de barras. Cada una de las barras dibujadas representa 'la línea de vida' de un número de Betti, en función del radio usado para generar los complejos. Puesto que estamos representando a los usuarios en dos dimensiones, solo pueden ser distintos de cero el primer y segundo número de Betti. Un ejemplo de código de barras asociado a un usuario se puede ver en la imagen 4.6.

Hemos seguido la metodología del TDA para generar estos códigos de barras asociados a cada espacio. Como ya hemos dicho, estos códigos de barras nos dan información sobre la estructura del espacio, pero lo que queremos en nuestro sistema de recomendación es comparar usuarios. Puesto que cada código de barras va asociado a un usuario, lo que queremos es comparar los códigos de barras. En el artículo abordado en la sección 2.1 sobre el modelado molecular, se

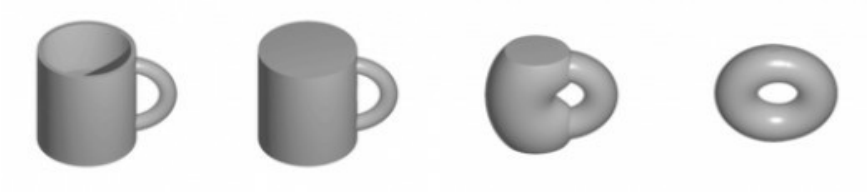


Figura 4.5: Taza y donut homeomorfos

Los diferentes pasos en la tranformación que establece un homeomorfismo entre una taza y un donut. Estos dos espacios, la taza y el donut, son homeomorfos.

utiliza un algoritmo para calcular la similitud entre dos códigos de barras. El algoritmo es el siguiente se muestra en la figura 4.7.

Este algoritmo calcula la Similitud basada en códigos de barras (Similarity Based on Bar-Codes) (S_{BOE}). Esta similitud tiene la siguiente fórmula:

$$S_{BOE}(A,B) = \begin{cases} S_{BO}(A,B) & \text{si } A \neq \varnothing \text{ and } B \neq \varnothing \\ 1 & \text{si } A = \varnothing \text{ and } B = \varnothing \\ 0 & \text{si } (A = \varnothing \text{ and } B \neq \varnothing) \text{ or } (A \neq \varnothing \text{ and } B = \varnothing) \end{cases} \tag{4.1}$$

donde A y B son dos códigos de barras, y

$S_{BO}(A,B) = \frac{1}{|A|+|B|} * [\sum_{a \in A} \sup_{b \in B} \frac{a \cap b}{a \cup b} + \sum_{b \in B} \sup_{a \in A} \frac{a \cap b}{a \cup b}]$

donde a y b denotan diferentes barras de los códigos de barras A y B, respectívamente.

El algoritmo realiza lo que se explica a continuación. Se calcula el índice de Jaccard para cada par de barras de los dos códigos de barras. Para cada barra de un código, se obtiene el índice de Jaccard mayor calculado con cada una de las barras del otro código. Se suman todos los supremos obtenidos y se divide por el número total de barras que contienen los dos códigos. Por otro lado, si uno de los códigos es vacío y el otro no, son totalmente distintos, por lo que se devolvera un 0. Y si ambos códigos son vacíos, ambos son idénticos y, por tanto, se devuelve un 1.

Gracias a estos procedimientos realizados, que son los que tiene la metodología del TDA, conseguimos obtener información sobre la estructura del espacio del que partimos. Recordemos que partimos de un espacio que representa la actividad que tiene un usuario con el sistema de recomendación, las puntuaciones que da a cada uno de los items que ha utilizado. Es decir, con este análisis podemos estudiar la forma que tiene la actividad del usuario sobre el sistema. De esta manera, estamos representando en un espacio bidimensional la actividad de un único usuario y vamos a comparar su estructura con la de otro usuario.

4.1.3. Implementación de la similitud

La implementación de la medida desarrollada se ha realizado en Matlab. Se ha elegido este lenguaje ya que la librería usada para la implementación de los códigos de barras está disponible, entre otros, en este lenguaje. Además, este lenguaje ha sido desarrollado para ser eficiente al realizar cálculos con vectores y matrices.

La librería utilizada para la implementación de los códigos de barras es JavaPlex. Esta librería está dedicada a la homología persistente de complejos simpliciales, con especial énfasis en la aplicación al TDA [11].

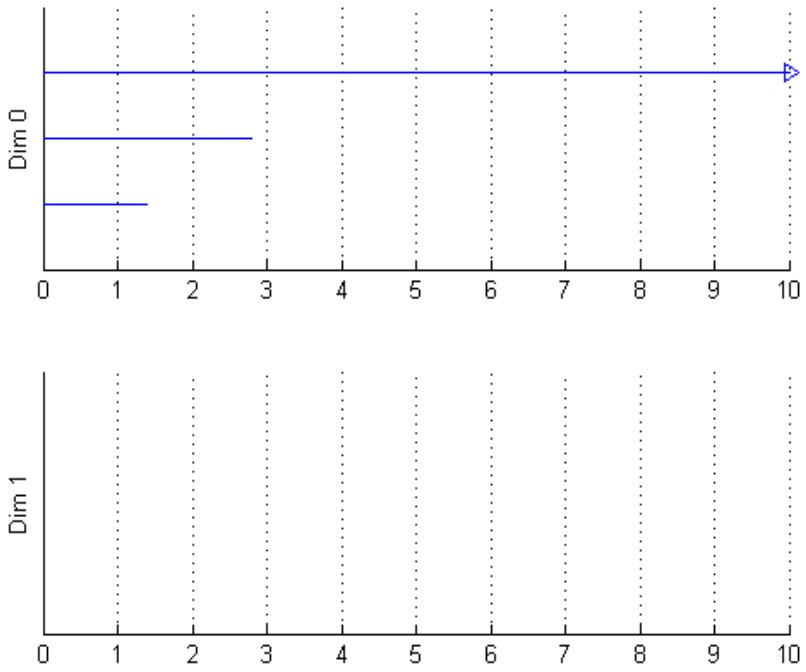


Figura 4.6: Código de barras asociado al usuario

Código de barras asociado al usuario descrito en la imagen 4.1. Empieza habiendo 3 componentes conexas. Cuando el radio llega a raíz de 2, pasa a haber solo dos componentes conexas, ya que se traza una arista entre el punto (1, 2) y (2, 3). Cuando el radio llega a raíz de 8, queda únicamente una componente conexa. No hay agujeros en el espacio.

Se ha hecho uso de esta librería ya que dispone de funciones que implementan cada uno de los pasos que se han de realizar hasta la obtención del código de barras, lo cual facilita el cálculo de nuestra similitud. De esta manera, la librería consta de una función que, dada una nube de puntos en $\mathbb{R}^N$, calcula el Vietoris-Rips Complex asociado a los datos. Se dispone de otra función que te proporciona los intervalos de homología persistente que nos indican los números de Betti. También ofrece una función que, a partir de los intervalos anteriormente citados, representa el código de barras asociado a la nube de puntos original.

Se comienza hallando el código de barras asociado a la nube de puntos que tenemos. La función que se encarga de esto recibe dos parámetros que afectan al cálculo de las barras del código. El primer parámetro es el valor máximo de la distancia a la que están dos puntos del complejo conectados por una arista. El segundo es el número de divisiones que va a realizar en el intervalo [0, x], siendo x el valor del primer parámetro, para las distancias entre los vértices de cada uno de los complejos que se van a hallar. Variando estos parámetros se obtendran diferentes resultados.

Tras esto, se procede a comparar estos códigos de barras que nos proporciona la librería. Se puede observar que el algoritmo explicado en la sección anterior en 4.7, puede llegar a ser muy ineficiente. La Operación Básica (OB) del algoritmo es la que está en la línea 12 (y 21). Suponiendo que el código de barras que contiene menos barras, contiene N barras, el algoritmo deberá realizar $O(N^2)$ OB's. El valor de N depende del número de puntos, de la disposición de los mismos y de los incrementos que se realizan en el radio de las circunferencias al calcular los complejos simpliciales. Teniendo en cuenta que el número de puntos es el número de items puntuados por el usuario, puede ser del orden de 500. Si N es 500, el algoritmo realizará

```
1: procedure S_BOE(A, B)
2:     if A = ∅ AND B = ∅ then
3:         return 1
4:     else if (A = ∅ AND B ≠ ∅) OR (A ≠ ∅ AND B = ∅) then
5:         return 0
6:     else
7:         pos ← 1
8:         for a ∈ A do          ▷ calculating the first sum from equation (3)
9:             jac[pos] ← 0
10:            for b ∈ B do
11:                if Jaccard(a, b) > jac[pos] then
12:                    jac[pos] ← Jaccard(a, b)
13:                end if
14:            end for
15:            pos ← pos + 1
16:        end for
17:        for b ∈ B do          ▷ calculating the second sum from equation (3)
18:            jac[pos] ← 0
19:            for a ∈ A do
20:                if Jaccard(a, b) > jac[pos] then
21:                    jac[pos] ← Jaccard(a, b)
22:                end if
23:            end for
24:            pos ← pos + 1
25:        end for
26:        sim ← 0
27:        for i ← 1, pos − 1 do                    ▷ averaging the results
28:            sim ← sim + jac[i]
29:        end for
30:        return sim/(pos − 1)
31:    end if
32: end procedure
33: procedure Jaccard(a, b)      ▷ Calculates the Jaccard index of two bars
34:    s ← a ∩ b
35:    u ← a ∪ b
36:    return |s|/|u|
37: end procedure
```

Figura 4.7: Algoritmo S_{BOE} [1]

aproximadamente 250000 OB's para comparar dos usuarios. Supongamos ahora que el número de usuarios es N también, aunque suele ser bastante mayor que el de items puntuados por un usuario. Si tenemos N usuarios, se han de calcular $\frac{N(N+1)}{2}$ coeficientes de similitud. Es decir, éste es el número de veces que se ha de ejecutar el algoritmo S_{BOE}. De esta manera, la complejidad del cálculo de la matriz de similitudes para nuestra similitud es de una complejidad de $O(N^4)$. cabe destacar que esta estimación de la complejidad se ha realizado como una cota inferior, ya que se ha tomado la N como el ínfimo del número de barras de ambos códigos, y además se ha tomado que es igual al número de usuarios.

La ineficiencia del algoritmo se encuentra en el cálculo del supremo de los coeficientes de Jaccard para cada una de las barras de los dos conjuntos. Para poder reducir la complejidad de esto, se puede observar la apariencia de un código de barras. En la imagen 4.8 se muestra un código de un usuario con muchas barras.

En la imagen 4.8 se puede observar que muchas de las barras del código son iguales. Esto se debe a que los valores de las puntuaciones de los usuarios son valores enteros del 1 al 5, lo cual da como resultado valores e intervalos muy repetitivos. De esta manera, a la hora de

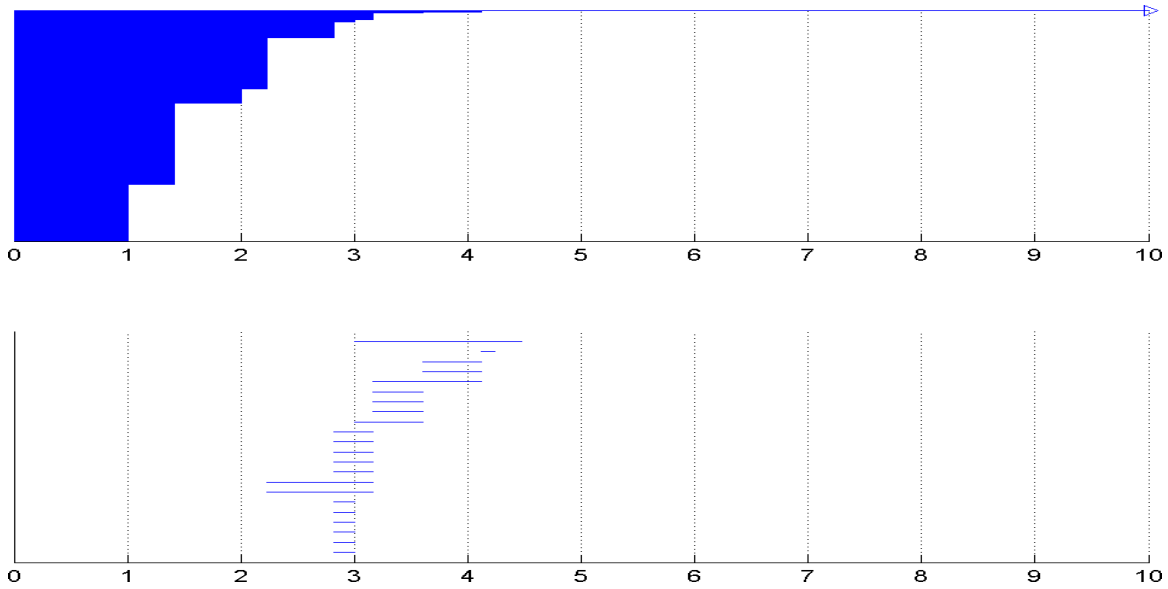


Figura 4.8: Código de barras de gran tamaño

calcular el supremo, estas barras idénticas intervienen una sola vez. Como las barras idénticas están colocadas de manera adyacente, se puede realizar una comprobación en los dos bucles para no realizar el mismo cálculo más de una vez. Esto reducirá el problema del cálculo de la similitud entre los dos códigos a calcular la similitud entre el subconjunto de las barras que solamente aparecen una vez en los códigos. Lo cual, disminuye considerablemente la complejidad del algoritmo.

4.1.4. Propiedades de los usuarios vistos como espacios topológicos

Ahora que se ha descrito cómo se calculan las similitudes entre dos usuarios siguiendo los pasos de este algoritmo, cabe preguntarse qué son dos usuarios similares vistos de esta forma. Empecemos recordando que la información extraída de la forma de los usuarios son los números de Betti, que permanecen invariantes en dos espacios topológicos homeomorfos. Es decir, dos usuarios serán idénticos si sus números de Betti asociados son los mismos exactamente para cada uno de los Complejos Simpliciales hallados. Esto puede verse de otro modo si retrocedemos a la idea de lo que son dos espacios topológicos homeomorfos. Dos espacios son homeomorfos si entre ellos dos existe un homeomorfismo. Dos ejemplos típicos de homeomorfismo son las traslaciones de puntos, las simetrías o los cambios de escala.

Supongamos que tenemos en el sistema a los usuarios 1 y 2 que han puntuado como lo indicado en la tabla 4.1.

	item 1	item 2	item 3	item 4	item 5	item 6	item 7	item 8
Usuario 1	2	3		1				
Usuario 2					2	3		1

Cuadro 4.1: Ejemplos de 2 usuarios de un sistema.

La tabla representa las puntuaciones otorgadas por cada usuario a cada uno de los 8 items que se muestran. Las celdas vacías representan que el usuario no ha puntuado dicho item.

Si representamos a los usuarios de la forma que se explica en la sección 4.1.1, nos quedan lo que se muestra en la imagen 4.9.

Si nos centramos en la forma que tienen cada uno de ellos, nos damos cuenta que el usuario

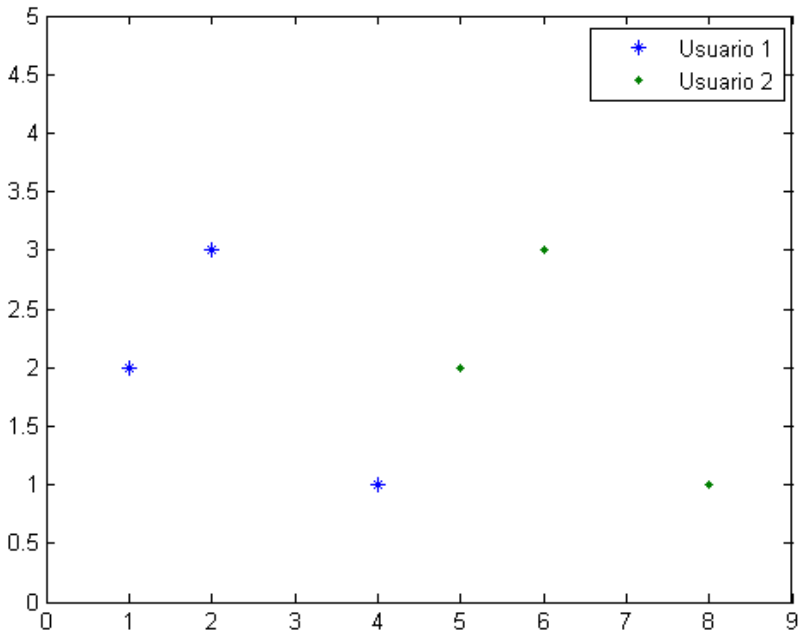


Figura 4.9: Ejemplo de dos usuarios representados

2 es la traslación de cada una de las puntuaciones del usuario 1. Es por esto, por lo que los códigos de barras quedan ambos idénticos, como el mostrado en la imagen 4.6.

De la misma manera, si se realiza una simetría o una dilatación o contracción de la forma que tiene el conjunto de puntos, se obtienen usuarios con códigos de barras similares o idénticos. Esto se puede apreciar en la figura 4.10

4.2. Implementación del sistema de recomendación

En esta sección se explica el diseño llevado a cabo para la implementación del sistema de recomendación que se ha usado para probar nuestra similitud. El sistema en su totalidad ha sido implementado en Matlab, al igual que nuestra similitud basada en espacios topológicos. En la figura 4.11 se muestra el diagrama de clases del sistema.

La clase **Recommender** es la base del sistema de recomendación. Es una clase abstracta, de la que heredarán todos los tipos de recomendadores que se podrían implementar. Tiene asociado principalmente un conjunto de datos que, para mayor modularidad, está implementado en la clase *DataSet*. Dispone de un método abstracto *score*, que tiene como objetivo predecir la puntuación que le daría un usuario a un item concreto.

La clase **KNNRecommender** hereda de la clase *Recommender*, e implementa un sistema de recomendación orientado en filtrado colaborativo, que usa el algoritmo de vecinos próximos para establecer vecindarios entre usuarios a la hora de recomendar. Con el método *calculateNeighbohood* se obtiene una matriz en la que se guarda para cada usuario, sus vecinos más próximos. Tiene asociada una similitud para calcular el parecido entre usuarios. Esta similitud está implementada en la clase *Similarity*. El algoritmo vecinos más próximos, obtiene para cada usuario

[1]Fuente: imagen tomada de [6]

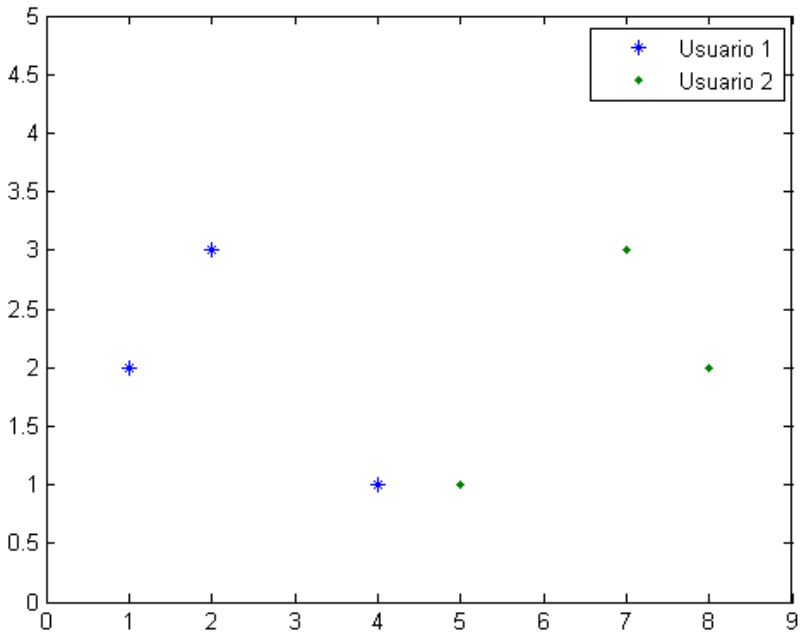


Figura 4.10: Ejemplo de dos usuarios simétricos

un número fijo k de vecinos, es decir, k usuarios más similares. Estos vecinos más similares se usan para calcular la puntuación de un item que no haya puntuado.

La clase **DataSet** es una clase abstracta que engloba la funcionalidad de manejar los datos del sistema, esto es, los usuarios, la matrix de puntuaciones, etc. Puesto que los datos se organizan de diferentes maneras dependiendo de la fuente de la que los obtengamos, se requieren unos métodos que puedan computar las diferentes formas de representación de los datos que pueda haber. En particular, nosotros obtenemos los datos de la empresa MovieLens, para la cual hemos implementado la clase MovieLensDataSet, que hereda de DataSet, que es capaz operar sobre los datos que recolecta la empresa. Por otro lado, cuando se prueba el sistema de recomendación, dividiendo el conjunto total de los datos en training y test, se requiere la existencia de otra clase que sea capaz de operar sobre un conjunto de test y así ofrecer las diversas funciones que se realizan sobre este conjunto de datos. Esta clase es **TestData**.

La clase **Similarity** es una clase abstracta de la que heredarán todas las similitudes implementadas. Dispone del método abstracto *calculateSimilarities* que implementará cada clase hija con su forma concreta de calcular la matriz de similitudes del sistema. Se han implementado cuatro similitudes diferentes, cada una en una clase que hereda de ésta: *CosineSimilarity*, *PearsonSimilarity*, *SBOESimilarity*. La clase **CosineSimilarity** implementa la similitud del coseno entre dos usuarios, donde su forma de calcular la similitud entre dos usuarios, es verlos como un par de vectores en $\mathbb{R}^N$ (donde N es el número de items en el sistema y cada componentes es la puntuación dada por el usuario al item correspondiente), y calcular el coseno entre los dos vectores. Este número que indica la similitud es un valor real entre 0 y 1. La clase **PearsonSimilarity** implementa la similitud de Pearson, donde su forma de calcular la similitud es verlos también como dos vectores en $\mathbb{R}^N$, como en la del coseno, y calcular el coeficiente de correlación de Pearson entre ellos. El coeficiente de similitud es un valor real entre -1 y 1.

Por otro lado, la clase **SBOESimilarity** implementa nuestra similitud basada en espacios topológicos. Para poder calcular el coeficiente de similitud entre dos usuarios en el método *calculateSimilarities*, hace uso de métodos intermedios estáticos para organizar el código. Inicialmente,

se guardan en una estructura *barCodes* todos los códigos de barras asociados a diferentes complejos simpliciales hallados sobre los espacios de puntos que representan los usuarios. Esto se hace, para que sea más eficiente su acceso, ya que si no habría que hallarlos varias veces. El método estático que halla el código de barras para los *Vietoris-Rips Complex* es *vietorisBarCode*. Tras esto, se procede a comparar los diferentes códigos de barras dos a dos, con el algoritmo S_{BOE} explicado en secciones anteriores. El algoritmo se ha dividido en varios métodos. El método *sboeAllDimensions* calcula el coeficiente S_{BOE} para todas las dimensiones del código de barras, esto es, para los números de Betti de todas las dimensiones que se indiquen. El método *sboe* calcula el coeficiente S_{BOE} para una dimensión en concreto, utilizando los métodos *sumSBOE*, que alberga el sumatorio de la fórmula del S_{BOE}, y el *jaccard* que calcula el coeficiente de Jaccard para dos barras de dos códigos.

La clase **Evaluator** es la que se encarga de evaluar si un recomendador funciona correctamente. Esto lo realiza calculando el error que comete en sus predicciones. Lleva asociada una instancia de la clase *TestData* que representa el conjunto de test de la prueba que se va a realizar, una clase *Recommender* que representa el recomendador a evaluar, y una clase *Metric* que representa la métrica que se va a utilizar para calcular el error en las predicciones.

La clase **Metric** es una clase abstracta que se encarga de calcular el error de predicción para un conjunto de test dado y un conjunto de datos que ha predicho un sistema de recomendación. Las clases que heredan de ésta implementan diversas fórmulas para medir el error en la predicción. La clase **RMSE** calcula el error con la fórmula del error cuadrático medio. Mientras que la clase **MAE**, lo calcula con la formula del error que es el valor absoluto de la diferencia de los real con lo predicho.

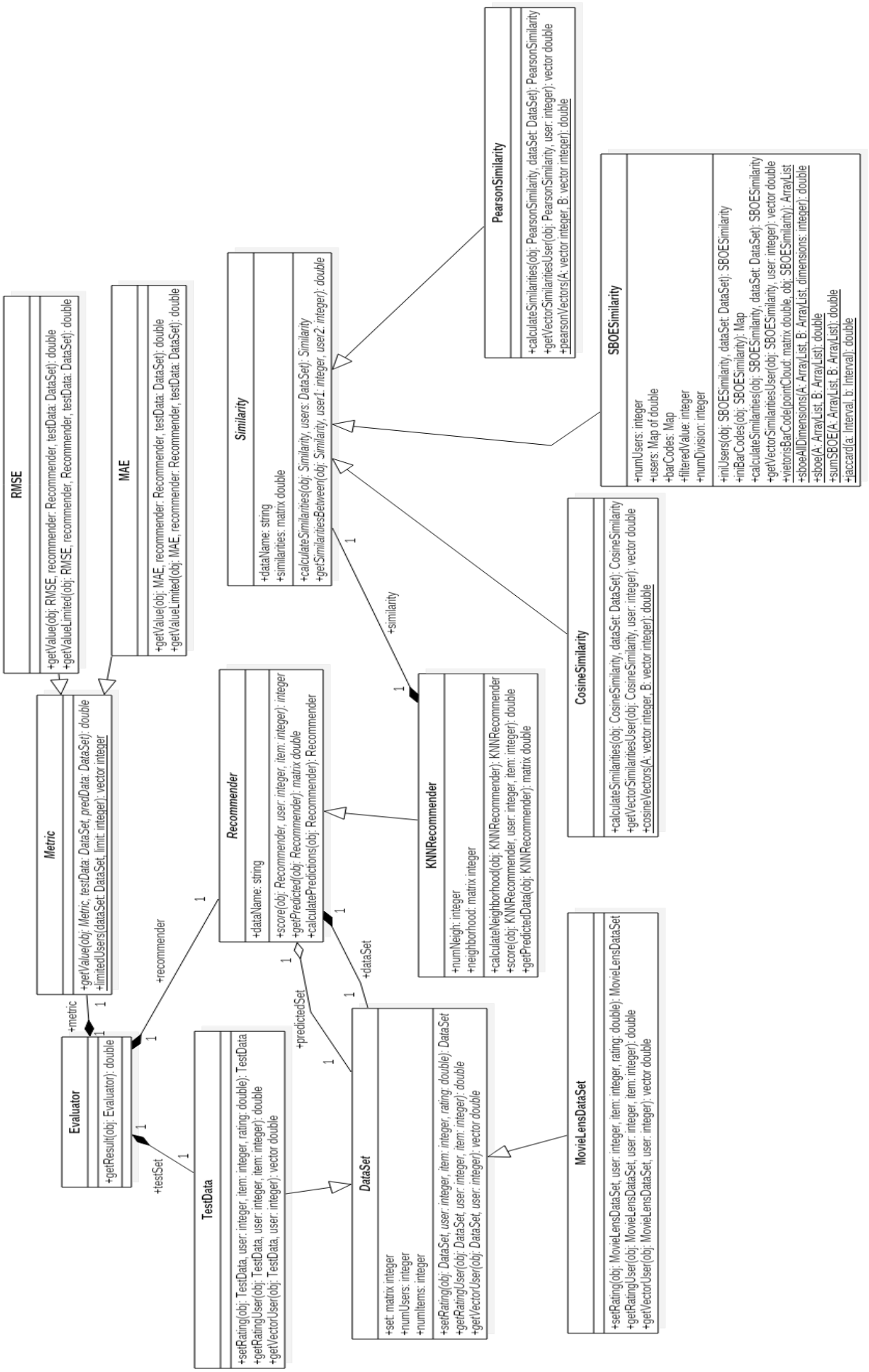


Figura 4.11: Diagrama de clases del sistema.

5

Experimentos y Resultados

En este capítulo se exponen los diferentes experimentos que se han realizado para probar la medida de similitud implementada. Se realiza, además, una explicación de los resultados obtenidos con respecto a otras métricas usuales. También se realiza un análisis del conjunto de datos sobre el que se trabaja.

5.1. Análisis del conjunto de datos

Realizar un análisis del conjunto de datos que se va a utilizar es el primer paso a la hora de probar el sistema de recomendación. La importancia de este análisis radica en la dependencia con los resultados finales que vayamos a obtener. Las características que vamos a estudiar son: qué tipo de items son los que recomienda el sistema, qué características tienen, y qué tamaño tiene el conjunto de ratings, usuarios e items.

Vamos a trabajar con un conjunto de datos denominado 'MovieLens 100k'. Este conjunto ha sido obtenido de la página de GroupLens [1]. En esta página se pueden conseguir diversos conjuntos de datos de diferentes tamaños. MovieLens es un sistema de recomendación basado en web que recomienda películas a los usuarios usando filtrado colaborativo. Fue creado en 1997 por GroupLens Research.

Este conjunto de datos contiene datos sobre ratings de películas del sistema de recomendación mencionado. Los datos recolectados de este tipo de sistemas guardan información principalmente de los usuarios registrados, los items con los que tratan y una lista de ratings que los usuarios han realizado sobre los items que han utilizado. Los usuarios y los items llevan un identificador asociado, y cada uno de los ratings relaciona un identificador de usuario con uno del item que ha puntuado y la correspondiente puntuación que le ha otorgado. Además, se suele guardar también información relacionada con los items y los usuarios del sistema. En nuestro caso, las películas del sistema tienen asociada una página web, una fecha y un género. Los usuarios tienen asociada una profesión, la edad y el sexo. Los ratings que guarda el sistema tienen asociada una puntuación del 1 al 5, y además guardan la fecha y hora en la que se ha realizado.

El conjunto de datos sobre el que vamos a trabajar está formado por 100000 ratings, 943 usuarios y 1682 items. En relación al tamaño que suelen tener los datos en este tipo de trabajos, es un conjunto pequeño. De hecho, esta misma empresa también dispone de conjuntos de datos de

mayores tamaños, con un millón e incluso veinte millones de ratings. Sin embargo, para la tarea de este trabajo que es comparar los resultados de recomendación de la distancia topológica frente a otras distancias más usuales, y así comprobar la precisión de este nuevo método, es preferible trabajar con este tamaño de los datos. De esta forma se minimiza el tiempo necesario para la realización de pruebas.

A continuación, vamos a analizar las características de la matriz de puntuaciones, para saber si las puntuaciones que han realizado los usuarios siguen alguna distribución específica. Para ello, se pueden representar los ratings de varias formas.

Se ha realizado un histograma en el que se representan el número de usuarios que han votado un número indeterminado de películas, que se muestra en la figura 5.1. Esto ha ayudado a ver que la mayor parte de los usuarios han votado un número pequeño de películas (menos de 100). Por lo que no tener en cuenta para algunas pruebas los usuarios que tengan un gran número de ratings, no hace que se pierda información sobre el conjunto de datos total. Además, esto puede aumentar el rendimiento de los programas de pruebas y que la tarea de obtener la precisión del método propuesto sea más rápida.

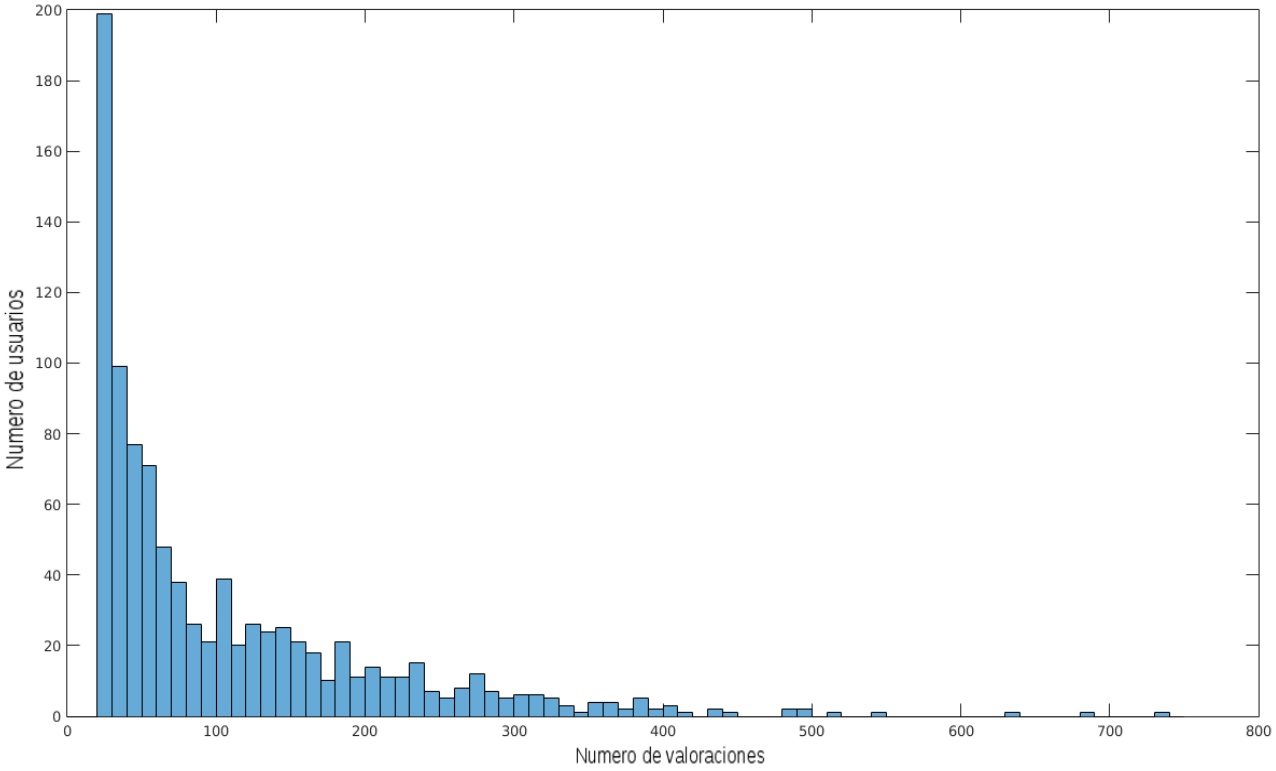


Figura 5.1: Histograma del conjunto de datos de MovieLens

En el histograma se puede observar que la mayoría de los usuarios tienen 50 o menos items puntuados.

5.2. Comparación con otras similitudes usuales

Puesto que hemos implementado una nueva manera de medir la similitud en sistemas de recomendación, la primera cuestión a tener en cuenta es la comparación con otras similitudes ya implantadas en sistemas de recomendación usuales. Para la realización de las pruebas, hemos

implementado, además de la similitud basada en códigos de barras, las similitudes basadas en el *coseno* y en *Pearson* para realizar dicha comparación.

La **similitud del coseno** representa a los usuarios como vectores en $\mathbb{R}^N$, siendo N el número de items del sistema. Cada componente del vector indica la puntuación dada por el usuario a ese item. Puesto que los usuarios no habrán puntuado todos los items del sistema, habrá componentes del vector que no tengan ninguna puntuación. De esta manera, para calcular el coseno entre los dos vectores que representan a los usuarios, se obtienen las componentes comunes a ambos, es decir, solo se realiza el coseno para las componentes de los items que han puntuado ambos. La fórmula de la similitud del coseno viene dada por:

$$sim(u,v) = \frac{\sum\limits_{\substack{i:r(u,i)\neq\varnothing \\ r(v,i)\neq\varnothing}} r(u,i)r(v,i)}{\sqrt{\sum\limits_{i:r(u,i)\neq\varnothing} r(u,i)^2 \sum\limits_{i:r(v,i)\neq\varnothing} r(v,i)^2}} \in [0,1]$$

donde $r(u,i)$ es la puntuación que le ha dado el usuario u al item i.

La **similitud de Pearson** representa a los usuarios como vectores en $\mathbb{R}^N$, al igual que en la del coseno. Se toman los vectores asociados a los usuarios como dos muestras de dos variables aleatorias, y se calcula el coeficiente de correlación de Pearson entre estas dos muestras. Este coeficiente representa una medida de la relación lineal entre los dos usuarios. La fórmula de este coeficiente es:

$$sim(u,v) = \frac{\sum\limits_{\substack{i:r(u,i)\neq\varnothing \\ r(v,i)\neq\varnothing}} (r(u,i)-\bar{r}_u)(r(v,i)-\bar{r}_v)}{\sqrt{\sum\limits_{\substack{i:r(u,i)\neq\varnothing \\ r(v,i)\neq\varnothing}} (r(u,i)-\bar{r}_u)^2 \sum\limits_{\substack{i:r(v,i)\neq\varnothing \\ r(v,i)\neq\varnothing}} (r(v,i)-\bar{r}_v)^2}} \in [-1,1]$$

donde $\bar{r}_u$ es la puntuación promedio del usuario u.

Mientras que la similitud del coseno tiene en cuenta la distancia euclídea de los usuarios vistos como vectores en $\mathbb{R}^N$, y la similitud de Pearson mide la correlación lineal entre dos variables aleatorias; la distancia topológica que proponemos se centra en la forma que tienen los usuarios. Representa a los usuarios como subespacios en $\mathbb{R}^2$, donde cada punto del espacio corresponde a la puntuación que le ha dado el usuario a cada item. Se representa la forma que tiene de puntuar a los items, transformando este conjunto de puntos de $\mathbb{R}^2$ en un código de barras. Puesto que cada usuario se representa con su respectivo código de barras, al final hay que realizar una comparación entre estos códigos de barras, una comparación entre la forma de puntuar de cada usuario. El algoritmo que nosotros usamos para comparar los códigos de barras es el que se muestra en la figura 4.7.

5.3. Determinación de la precisión del algoritmo

En un sistema de recomendación disponemos de un conjunto de datos recolectado con información sobre la actividad de cada uno de los usuarios. Cada vez que se necesita recomendar un item nuevo a un usuario, se calcula la predicción del rating que dicho usuario le daría a cada uno de los items que aún no ha puntuado, y le recomienda la que mayor puntuación obtenga.

De esta manera, para poder probar la precisión del sistema de recomendación, hemos dividido el conjunto de datos del que disponemos en dos subconjuntos: el conjunto de training y el de test. Nuestro sistema solamente va a disponer del conjunto de training, es decir, no va a saber que existe un conjunto de test que vamos a usar para comparar con los resultados que estima. Por ello, va a predecir la puntuación que le daría a cada item que no haya puntuado cada usuario con las similitud que sea asignada. Puesto que tenemos el subconjunto de test que contiene datos reales de las puntuaciones de los usarios pero que no han sido mostradas al sistema, podemos comparar estos valores con los que ha predicho el sistema, calculando así el error de predicción.

Nuestro conjunto de datos dispone de conjuntos diferenciados de entrenamiento y de test para el sistema de recomendación. Más adelante se explicará cómo se emplean estos conjuntos. En concreto, se dispone de 7 pares de archivos de training y test, que dividen el conjunto total de datos en distintas proporciones. Estos archivos son de u1 hasta u5, y ua y ub. Estos archivos simplemente contienen diferentes maneras de particionar el conjunto de datos de partida. A la hora de comparar resultados, solamente se van a usar los conjuntos de u1 hasta u5 ya que éstos contienen el mismo número de datos para entrenamiento y test.

La forma de comparar los errores cometidos, se puede realizar con distintas métricas que miden el error cometido. En nuestro caso, una de las métricas que hemos empleado ha sido el error medio absoluto **MAE**, que viene dado por la fórmula:

$$MAE = \frac{1}{|test|} \sum_{(u,i) \in test} |\hat{r}(u,i) - r(u,i)| \ , \ 0 \leq MAE \leq max(r)$$

donde *test* es el conjunto de test, $|test|$ es el número de elementos del conjunto de test y $\hat{r}(u,i)$ es la puntuación del usuario u al item i predicha por el sistema y $r(u,i)$ la real.

Esta métrica calcula la distancia que hay entre cada predicción y su valor real, tomada como un valor absoluto de la diferencia; y después realiza la media de estas diferencias. Por otro lado, la métrica **RMSE** mide la distancia realizando el cuadrado de la diferencia:

$$RMSE = \sqrt{\frac{1}{|test|} \sum_{(u,i) \in test} (\hat{r}(u,i) - r(u,i))^2}$$

5.3.1. Eficacia del algoritmo

La similitud calculada mediante los códigos de barras asociados a cada usuario, puede calcularlos con mayor o menor precisión en función de unos parámetros, como se mencionó en el capítulo anterior. El primer parámetro es el valor máximo de la distancia a la que están dos puntos del complejo conectados por una arista. Este parámetro puede determinar que al final todos los puntos estén conectados con todos los demás. También va a determinar el tamaño máximo de las barras del código. El segundo es el número de divisiones que va a realizar en el intervalo [0, x], siendo x el valor del primer parámetro, para las distancias entre los vértices de cada uno de los complejos que se van a hallar. Por otro lado, este parámetro determina la exactitud en el tamaño de cada barra. Ya que una barra puede tener un tamaño igual a un número irracional, que requiera de pequeños incrementos para que tenga mayor precisión en el cálculo. Variando estos parámetros se obtendran diferentes resultados.

Se han realizado diferentes pruebas del sistema de recomendación con la similitud propuesta variando el valor de estos parámetros. De esta manera, se ha calculado el valor del error para cada una de las dos métricas en función de estos dos parámetros.

Tal y como se ha explicado en la sección 5.1, el cálculo de códigos de barras con muchos puntos puede resultar ineficiente. Por esto, hemos simplificado el conjunto de los datos, viendo que la mayor parte de los usuarios ha puntuado un número reducido de items. Por ello, no se pierde demasiada información tomando solo usuarios con pocos ratings. Es por esto por lo que se han hecho las pruebas del sistema haciendo variar el conjunto de datos de partida, usando solamente usuarios que han puntuado menos de un número fijo de items.

	RMSE	MAE
S_{BOE} 10 10	1.6011	1.214
S_{BOE} 50 20	1.553	1.1764
S_{BOE} 100 15	1.5364	1.1636
S_{BOE} 1000 15	1.6254	1.2319

Cuadro 5.1: RMSE y MAE para diferentes valores de parámetros en la similitud basada en S_{BOE} para usuarios que han puntuado como mucho 50 items, para 100 vecinos en KNN.

En la tabla 5.1 se muestra el RMSE y el MAE resultante de ejecutar el sistema de la similitud basada en el algoritmo S_{BOE}, para el archivo u1 con 100 vecinos.

Además, es interesante ver si la similitud funciona mejor o peor cuantos más vecinos utilice el algoritmo de KNN. De esta manera, los diferentes resultados y gráficas sobre los errores, se han tomado en función del número de vecinos que utiliza KNN.

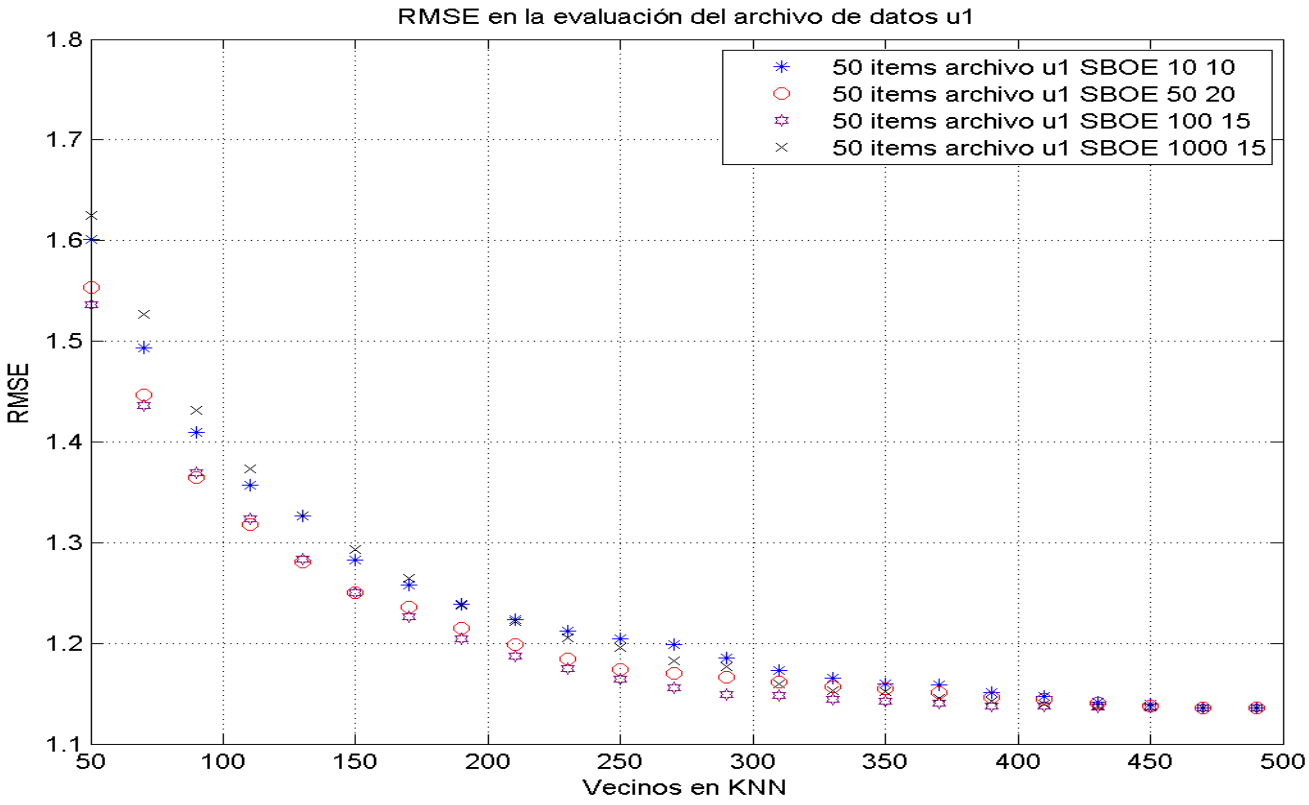


Figura 5.2: Gráfica de la similitud basada en S_{BOE} para el archivo u1 utilizando usuarios con 50 items puntuados o menos. La similitud basa en S_{BOE} usa los dos parámetros descritos anteriormente con un valor igual a los dos últimos números de cada gráfica.

Como se puede observar en la imagen 5.2, los errores comienzan un poco altos, en torno a 1.5. Pero cuando los vecinos usados por el sistema aumentan, el error decrementa bastante, hasta

llegar casi a 1.1. Los mejores resultados, con variaciones pequeñas, se dan para los parámetros 50 y 20, y para 100 y 15. Esto se debe a que se realizan más divisiones en la distancia usada entre los vértices en los complejos. Así, la longitud de las barras se obtiene con una precisión mayor. En esta imagen, y en las posteriores solamente se muestran resultados sobre la ejecución del archivo u1, esto se debe a que los resultados de los 5 archivos son muy similares, y solo mostramos este como valor representativo. De la misma manera, las gráficas del RMSE y el MAE tienen un comportamiento similar, por lo que solamente vamos a mostrar gráficas del RMSE.

Para un mayor conocimiento sobre el significado del error obtenido para nuestra similitud, podemos compararlo con el que obtienen similitudes usuales. Para ello, hemos ejecutado el sistema de recomendación con la similitud del coseno y la de Pearson, haciendo variar el número de vecinos que usa KNN. Al igual que para nuestro algoritmo, también las hemos probado para el conjunto de datos reducido. Esto es, tomando solamente usuarios que tengan un número de puntuaciones menor o igual que x items.

En las tablas 5.2 y 5.3 se pueden ver los errores medidos con RMSE y MAE para todas las similitudes probadas juntos con las dos que mejor resultado han dado de las basadas en el S_{BOE}.

	RMSE	MAE
S_{BOE} 50 20	1.553	1.1764
S_{BOE} 100 15	1.5364	1.1636
coseno	1.1455	0.89964
Pearson	1.2345	0.94907

Cuadro 5.2: RMSE y MAE para todas las similitudes probadas para usuarios que han puntuado como mucho 50 items, para 100 vecinos en KNN.

	RMSE	MAE
S_{BOE} 50 20	1.1616	0.90138
S_{BOE} 100 15	1.1482	0.89376
coseno	1.0791	0.86692
Pearson	1.5728	1.1298

Cuadro 5.3: RMSE y MAE para todas las similitudes probadas para usuarios que han puntuado como mucho 50 items, para 300 vecinos en KNN.

En la figura 5.3 pueden verse representadas la similitud del coseno y la de Pearson para todos los usuarios del sistema. La similitud del coseno parece dar bastante mejor resultado que la de Pearson. Esto puede deberse a que, como trabajamos con un conjunto de datos relativamente pequeño, los coeficientes de correlación de Pearson entre los usuarios salen negativos con mayor probabilidad. Esto hace que la similitud de Pearson de unos errores mayores cuando se usan pocos y muchos vecinos. En la figura 5.4, se representa lo mismo, pero usando solamente usuarios que hayan puntuado 50 items o menos. En ésta, se aprecia algo similar a lo que ocurre utilizando todos los usuarios.

Finalmente, en la figura 5.5 se puede apreciar la comparativa entre las similitudes de Pearson y del coseno y de las que daban mejores resultados de las basadas en S_{BOE}. Inicialmente, tanto la de Pearson como las dos S_{BOE} empiezan con un error en torno a 1.5, algo peor que la del coseno, que empieza con 1.2 aproximadamente. Sin embargo, según avanzan los vecinos, la similitud de Pearson empeora su precisión. Esto se debe al gran número de usuarios que dan coeficientes de correlación negativos. Mientras que las otras tres similitudes van mejorando su error hasta alcanzar una tasa de 1.1.

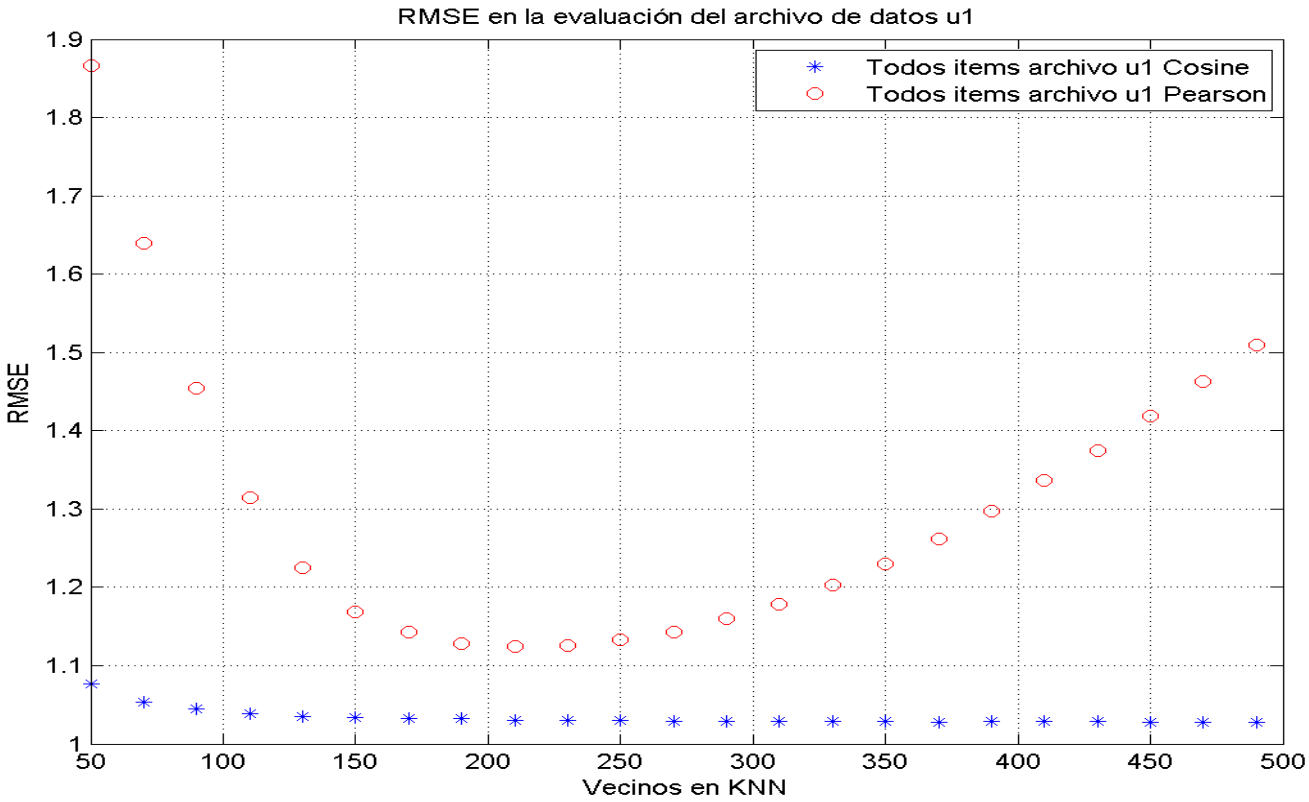


Figura 5.3: Gráfica de la similitud de Pearson y del coseno para el archivo u1 utilizando todos los usuarios

5.3.2. Eficiencia del algoritmo

La eficiencia de un algoritmo es muy importante, sobre todo si se trata de algoritmos que deben dar respuestas en tiempo real. Un sistema de recomendación puede necesitar predecir la recomendación a un usuario en un tiempo limitado. La actividad que realiza puede dividirse en dos pasos:

1. El cálculo de la matriz de similitudes entre todos los usuarios. En ciertos casos suele ser más eficiente tener calculada esta matriz, e ir actualizándola a medida que los usuarios interactúan con el sistema, en lugar de calcular cada coeficiente cuando es necesario.

2. Suponiendo que se tiene precalculada la matriz de similitudes, el siguiente paso sería usar un algoritmo de recomendación para predecir qué puntuación le daría un usuario a un item. Este algoritmo en nuestro caso es el KNN.

Si dividimos la actividad del sistema de esta manera, la implementación de una nueva similitud supondría un cambio en la eficiencia del primer paso. Es por ello, por lo que nos centramos en el tiempo que consume el sistema en calcular la matriz de similitudes.

Para obtener una estimación del tiempo, hemos calculado la media de lo que tarda en calcularse un coeficiente de la matriz de similitudes, es decir, calcular la similitud entre un usuario y otro. Y por otro lado, hemos estimado el tiempo medio que se tarda en calcularse una fila de la matriz de similitudes, ya que es lo necesario para recomendar un item a dicho usuario.

Nuestro algoritmo basado en S_{BOE} puede dividirse, a su vez, en dos pasos cuando calcula el coeficiente de similitud entre dos usuarios:

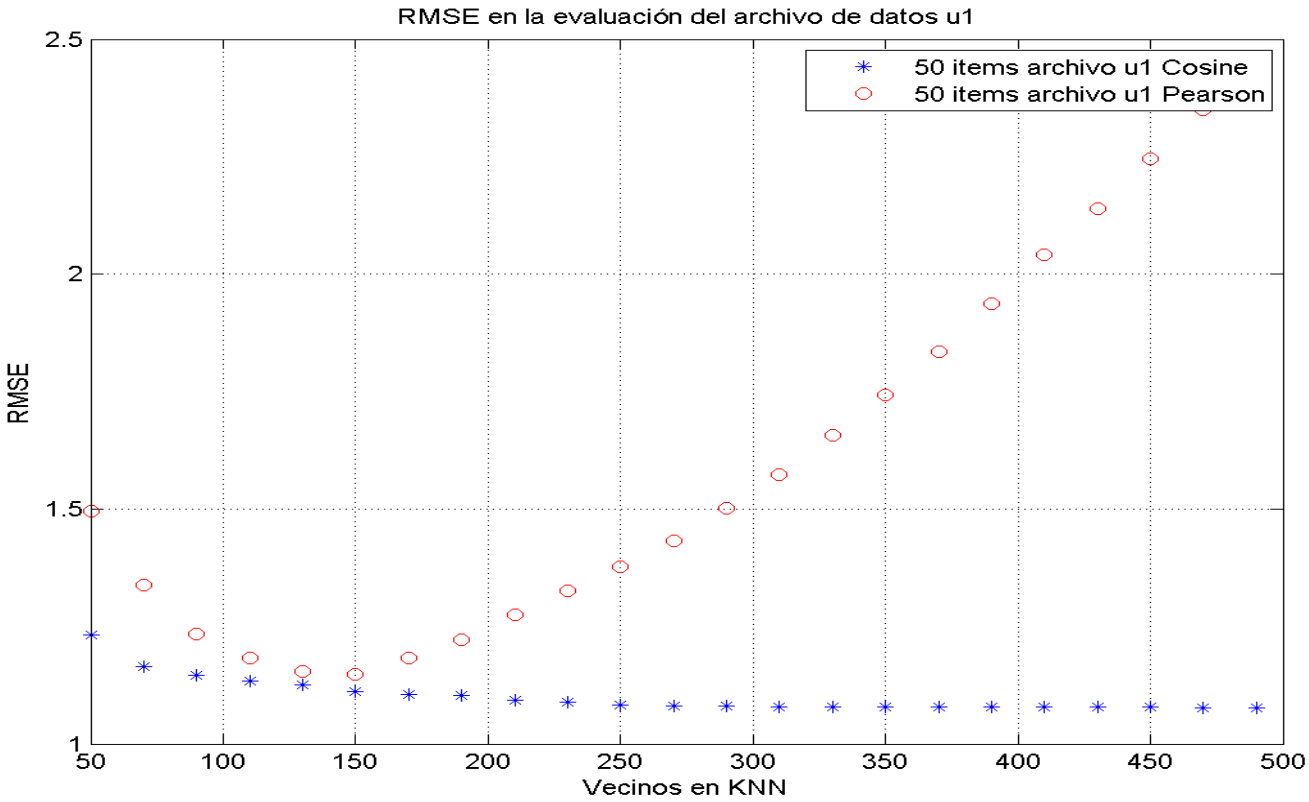


Figura 5.4: Gráfica de la similitud de Pearson y del coseno para el archivo u1 utilizando usuarios que han puntuado 50 items o menos

1. El cálculo de los códigos de barras asociados a estos dos usuarios.
2. El coeficiente de similitud entre ambos códigos, usando el algoritmo S_{BOE}.

	Código de barras	S_{BOE}
Media para un par de usuarios	0.07928	0.003666
Media para un usuario con todos los demás	30.047	1.3894

Cuadro 5.4: Eficiencia del algoritmo S_{BOE}

En la tabla 5.4 se pueden ver los resultados de eficiencia del algoritmo. De los dos pasos del algoritmo, se aprecia que la parte que más tarda es la del cálculo de los códigos de barras. Esto se debe a que, si se obtiene el coeficiente de similitud que hay entre un usuario y todos los demas, es necesario obtener el código de barras asociado a todos los usuarios del sistema. Es decir, para el caso de calcular la matriz entera, este tiempo sería el mismo, ya que habría que calcular todos los códigos de barras también.

Para tener una comparativa de los resultados de eficiencia, se han obtenido también los rendimientos para las similitudes del cosneo y de Pearson.

	Coseno	Pearson
Media para un par de usuarios	0.00014811	0.00040258
Media para un usuario con todos los demás	0.056134	0.15258

Cuadro 5.5: Eficiencia de coseno y Pearson

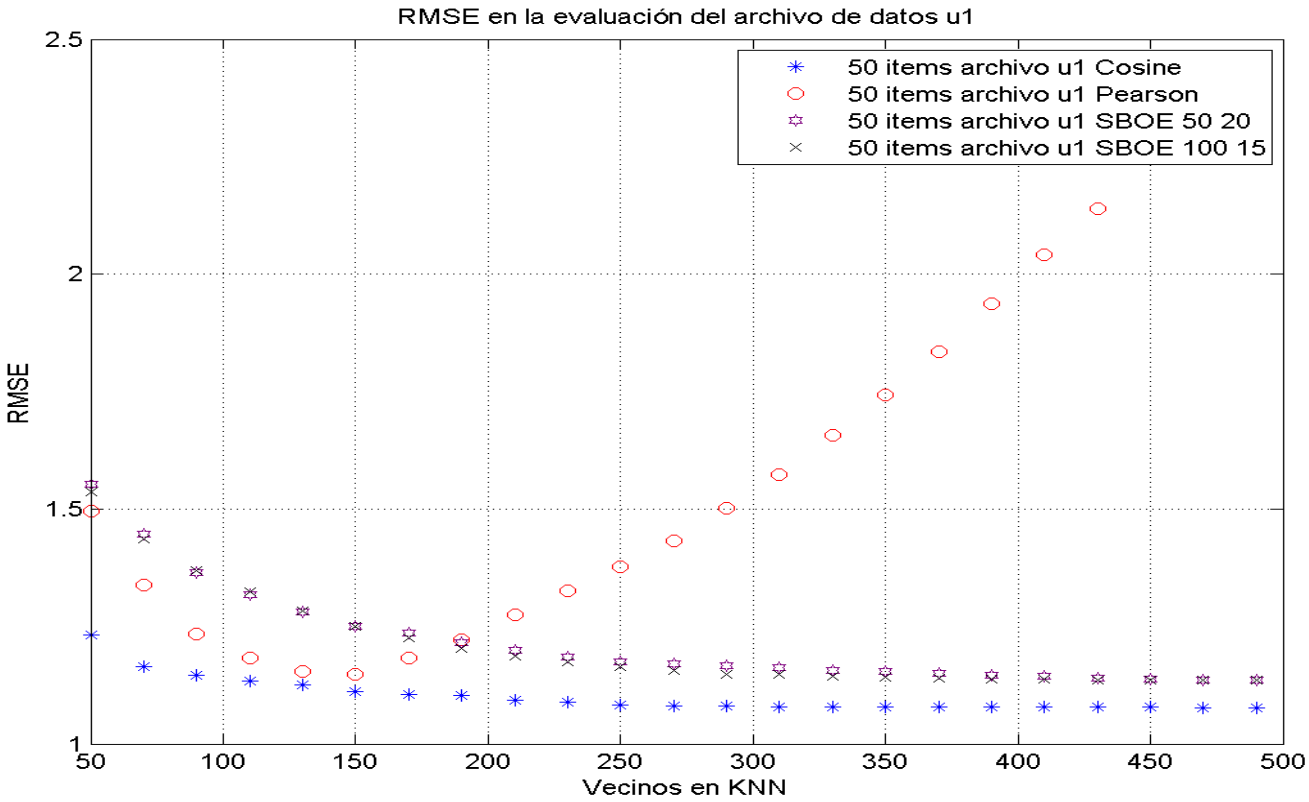


Figura 5.5: Gráfica de la similitud con las tres similitudes para el archivo u1 utilizando usuarios que han puntuado 50 items o menos. La similitud basa en S_{BOE} usa los dos parámetros descritos anteriormente con un valor igual a los dos últimos números de cada gráfica.

Los resultados se pueden ver en la tabla 5.5. Entre ellas dos, se puede observar que la similitud del coseno es mas eficiente, ya que el cálculo del coseno entre dos vectores es una operación muy sencilla. Si las comparamos con los resultados de nuestra similitud de la tabla 5.4, observamos que ésta da peores resultados. Puesto que los códigos de barras van asociados a cada usuario y se calculan todos, es más justo compararlos con el tiempo de ejecución del algoritmo S_{BOE}. En este caso, tarda también más que el coseno y Pearson, ya que realiza muchas más operaciones entre todas las barras.

6

Conclusiones y trabajo futuro

El trabajo realizado ha conllevado la realización de muchas tareas diferentes relacionadas con las asignaturas impartidas a lo largo del doble Grado en Ingeniería Informática y Matemáticas. La investigación llevada a cabo ha profundizado en temas sobre la topología para poder entender y llevar a cabo toda la teoría que hay detrás de la similitud implementada. Este estudio se ha centrado en el cálculo de distintos complejos simpliciales para poder calcular diferentes números de Betti y así después poder representarlo en un código de barras. Estas técnicas ya han sido llevadas a cabo en otros ámbitos y, por tanto, solamente ha habido que estudiarlas e implementarlas. Por otro lado, se ha realizado un estudio de lo que es un sistema de recomendación y de los distintos tipos existentes hoy en día.

Sin embargo, el objetivo de este trabajo era el estudio y posterior desarrollo de posibles maneras de implementación de una similitud que usara técnicas relacionadas con la topología, como las que realiza la metodología del TDA. Es decir, estudiar la relación entre la topología computacional y los sistemas de recomendación que usan filtrado colaborativo. Por ello, la tarea más importante posiblemente haya sido la transformación del problema de la recomendación en uno que se incluya en el contexto de la topología computacional; y, posteriormente, la interpretación de los resultados obtenidos tras aplicar estas técnicas, volviendo a la recomendación a usuarios.

Para poder llevar a cabo este Trabajo de Fin de Grado ha sido necesario el empleo de técnicas muy diversas, como el estudio de los conceptos teóricos de lo que es una topología y de la topología algebraica aplicada a la programación, programación de un sistema de recomendación, programación de algoritmos para la realización de las distintas similitudes, manipulación de datos, etc.

6.1. Conclusiones obtenidas

La topología computacional y más en concreto el TDA, trata de extraer información de los conjuntos de datos, que normalmente otros análisis no tienen en cuenta, además de simplificar en muchos casos el problema a abordar. Puesto que en los sistemas de recomendación actuales cada vez se dispone de más información sobre los usuarios y cada vez los conjuntos de datos son de mayor tamaño, surge la necesidad de un estudio sobre la aplicación de estas técnicas a la

tarea de recomendar. Es por ello, por lo que se ha realizado la implementación de una similitud basada en el TDA para un sistema de recomendación que usa filtrado colaborativo.

Puesto que el TDA es una técnica que ha surgido en los últimos años, no hay un gran número de ámbitos en los que se haya puesto en práctica. Por ello, hay una gran área de investigación que aún no se ha abordado para el análisis de datos a gran escala, así como un desarrollo a nivel teórico de los problemas que pueden abordar estas técnicas de topología.

Al igual que con las similitudes usuales que usan los sistemas de recomendación, el problema crucial es cómo vemos los usuarios del sistema. Tanto el coseno como el coeficiente de correlación de Pearson, usados a la hora de comparar resultados en capítulos anteriores, se basan en conceptos matemáticos básicos que tienen su significado y utilidad en el contexto de una estructura matemáticas concreta. El paso que hay que realizar para poder aplicarlos es ver los usuarios como dos vectores a los que aplicar el coseno o como dos distribuciones de probabilidad a los que aplicar el coeficiente de Pearson. En resumen, el paso más complicado e importante a realizar es enfocar a los usuarios de distintas maneras en las que poder aplicar un concepto matemático con el que poder compararlos. En nuestro caso, este paso ha sido el de ver a los usuarios como un espacio topológico bidimensional, para después poder aplicar las medidas de similitud entre espacios, basadas en los números de Betti y códigos de barras asociados.

Tras una comparación con las medidas que suelen usarse en los sistemas actuales, se han obtenido resultados de peor eficacia. Puesto que es el primer acercamiento que se realiza a la aplicación de estas técnicas a estos ámbitos, puede estudiarse en mayor profundidad cómo mejorarlos, ya que pueden combinarse con otra información del conjunto de datos.

La eficiencia del algoritmo llevado a cabo ha resultado peor que la de otros algoritmos que realizan cálculos más simples sobre los datos. No obstante, si se consiguen obtener resultados que sean mejores que los de estos otros algoritmos, podría ser de utilidad en sistemas donde la recomendación a tiempo real no sea importante. Además, puesto que la matriz de similitudes del sistema puede estar precalculada o puede ser recalculada cada vez que un item o usuario se da de alta en el sistema, el tiempo que se tarda en recomendar un item a un usuario será el mismo que el de otras similitudes, puesto que el algoritmo KNN (o otro similar) sería el mismo. Es decir, todos los algoritmos utilizados para recomendar en un sistema colaborativo, no tienen porqué ser conscientes de cómo se ha calculado la matriz de similitudes sobre la que operan.

6.2. Trabajos futuros

Tras el desarrollo de este trabajo se han planteado diversas nuevas incógnitas que dan pie a posibles trabajos futuros que parten de lo ya realizado.

1. Todo lo que se ha desarrollado para el trabajo sirve para dar pie a seguir explorando sobre otras posibles técnicas de la topología computacional aplicadas al manejo de datos de los sistemas de recomendación. Ya no solo que tengan que ver con las topología algebraica basada en códigos de barras que se ha estudiado para el desarrollo de esta similitud.

2. Como ya hemos resaltado en la sección anterior, probablemente el paso más importante en este análisis de los datos, es el cómo se ven los usuarios en este caso como espacios topológicos. En nuestra similitud, los hemos visto como espacios bidimensionales, donde el 'eje x' representaba el item y el 'eje y' representaba la puntuación otorgada a dicho item. Otra posible implementación, sería reordenar los items del espacio para agruparlos por géneros. Podría representarse en un tercer eje el género de las películas, y así ser un espacio tridimensional. De esta manera, podrían añadirse más propiedades de los items existentes

en más dimensiones. El problema de calcular los números de Betti puede aplicarse a tantas dimensiones como se quiera, haciendo uso de las mismas técnicas, por lo que esto no sería un problema.

3. El algoritmo de comparación de los códigos de barras obtenidos, S_{BOE}, se ha usado en otros ámbitos en los que se ha aplicado el TDA, como en [6]. Pero puede encontrarse un algoritmo que de mejores resultados que éste, tanto en el cálculo de la similitud como en el tiempo de ejecución. Este algoritmo puede ser independiente de la forma de calcular los complejos simpliciales y los números de Betti, así como de las dimensiones del espacio en el que nos encontremos. Es decir, es independiente del problema planteado en el punto anterior.

4. En este trabajo se han comparado los resultados con tan solo otras dos similitudes: la de Pearson y la del coseno. Pero podrían utilizarse otras que suelan usarse, para obtener una validación mejor de los resultados. De la misma manera, se podrían usar otras métricas del error. O incluso, otras métricas que no se centren en el error de predicción, como utilizar métricas de ránking: P@k, nDCG, MAP, etc.

Bibliografía

[1] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, December 2015.

[2] Afra Zomorodian. Topological data analysis. *Advances in applied and computational topology*, 70:1–39, 2012.

[3] Jónathan Heras, María Poza, Maxime Dénès, and Laurence Rideau. Incidence simplicial matrices formalized in coq/ssreflect. *Intelligent Computer Mathematics*, pages 30–44, 2011.

[4] Topological data analysis for the working data scientist - anthony bak. `https://www.youtube.com/watch?v=3Z73Wd2T1xE&list=PLvzaBsnVmPpCZfUlTzpAt59gRAEvqUQfZ`. Último acceso 27/06/2017.

[5] Tamal K Dey, Herbert Edelsbrunner, and Sumanta Guha. Computational topology. *Contemporary mathematics*, 223:109–144, 1999.

[6] Gabriell Máté, Andreas Hofmann, Nicolas Wenzel, and Dieter W Heermann. A topological similarity measure for proteins. *Biochimica et Biophysica Acta (BBA)-Biomembranes*, 1838(4):1180–1190, 2014.

[7] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Introduction to Recommender Systems Handbook*, pages 1–35. Springer US, Boston, MA, 2011.

[8] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[9] James R Munkres. *Elements of algebraic topology*, volume 2. Addison-Wesley Menlo Park, 1984.

[10] Robert Ghrist. Barcodes: the persistent topology of data. *Bulletin of the American Mathematical Society*, 45(1):61–75, 2008.

[11] Andrew Tausz, Mikael Vejdemo-Johansson, and Henry Adams. JavaPlex: A research software package for persistent (co)homology. In Han Hong and Chee Yap, editors, *Proceedings of ICMS 2014*, Lecture Notes in Computer Science 8592, pages 129–136, 2014. Software available at `http://appliedtopology.github.io/javaplex/`. Último acceso 27/06/2017.